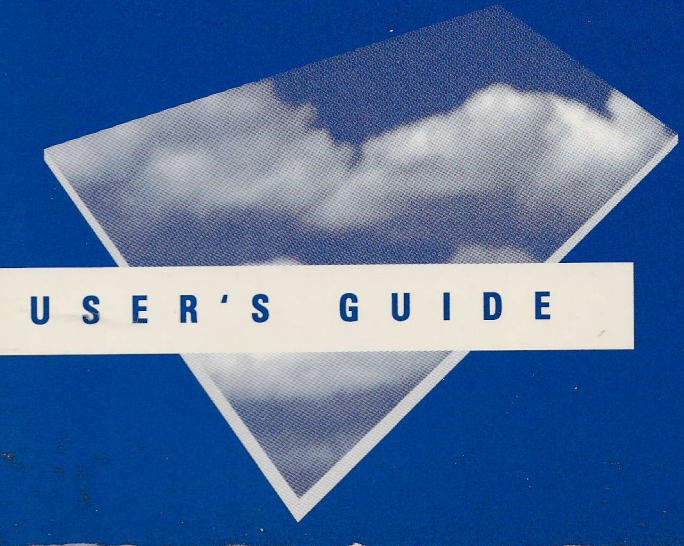
ADVANCED REVELATION®



USER'S GUIDE

RevelationTechnologies

Advanced Revelation®

Version 3.0

User's Guide

SER # 35474

COPYRIGHT

TRADEMARK NOTICE

SOFTWARE COPYRIGHT NOTICE

Advanced Revelation Version 3.0
October 1992

Getting started

Table of contents

1. About Advanced Revelation

Welcome to Advanced Revelation! We're pleased that you're joining the many developers and users who have found that Advanced Revelation is the most powerful and flexible development environment available today for PCs.

What you can do with Advanced Revelation

What can you do with Advanced Revelation? The short answer is: almost anything you can think of. Advanced Revelation is called an "application development environment" because you can use it to create complete applications that help users perform their jobs more efficiently.

But the definition of "application" can be quite broad. This includes, of course, the many types of business solutions that involve collecting and managing data (database functions): everything from accounting and customer tracking to more exotic applications such as forecasting and modeling and even real-time process control.

But what you can do with Advanced Revelation is limited only by your imagination. Examples of the range of applications currently running includes graphics (displaying photos on the screen), direct-to-press publishing, word processing systems, world-wide brokerage systems, personnel tracking with interfaces to payroll systems on mainframes, and creating bar-code-based tracking systems.

What all of these applications have in common is that their developers selected Advanced Revelation over all other packages available. For an idea of why they chose Advanced Revelation, and why they were able to develop such a diversity of applications, read on.

Features of Advanced Revelation

Advanced Revelation includes a full array of data management tools. Listed below is an overview of the many features that you will be working with in Advanced Revelation. You can find details about each of these features later in this documentation set.

Online help In addition to this documentation set, Advanced Revelation includes extensive online help. Simple keystrokes virtually everywhere in Advanced Revelation teach you more about the current process or displays options.

Database Management Featuring the most flexible database management system available today, Advanced Revelation puts almost no limits on how large your tables, rows, and columns can be. Advanced Revelation's dictionary-driven database management system stores all information about a data table in a central dictionary that can be changed at any time.

Application Development To create applications, you can use the most innovative development tools available. For example, to create a window, you simply "paint" prompts on the screen—Advanced Revelation will even create the data table for you at the same time.

Not only are the tools easy to use, but they are powerful as well. You can change any aspect of the application by simply editing its "template". In addition, you can add custom processing at any point in the application with little or no programming, saving hours or days of development time.

Reporting You can create almost any type of report that you require. Tools are provided to create everything from simply columnar reports to powerful mail merges. You can even use the window tools to "paint" a report on the screen.

SQL In addition to its other database management tools, Advanced Revelation includes a full ANSI-compliant SQL that can be used with any data in your application, including data available with Environmental Bonds (see below).

Networking If you plan to run applications on a local area network, Advanced Revelation is ready to support all major networks. Advanced Revelation also includes a wide range of options for coordinating multi-user access to data.

Environmental Bonding One of the most innovative capabilities in any database manager, Environmental Bonding allows you full access to data stored in other formats or other locations. For example, your application can read and write data in dBASE III files, ASCII files, files on a database server, or in a host of other environments.

Programming For applications that require programming, Advanced Revelation makes available R/BASIC, the best database programming language available today. Featuring dozens of functions and commands especially suited to database programming, R/BASIC also features user-defined functions, dynamic linking, an interactive debugger, and other sophisticated capabilities.

How Advanced Revelation works

If you have used other environments to create applications in the past, you might find working in Advanced Revelation to be quite different. Although you do many of the same things that you do in other environments—create and maintain tables of data, build windows and menus, design and run reports, perhaps write programs—both the way you build the application and the way that you run it are likely to be new to you.

Here are just a few of the features that distinguish Advanced Revelation:

Interactive environment

To use Advanced Revelation, you must first *log on*. This invokes the Advanced Revelation environment. From that point, Advanced Revelation is in control, waiting for your commands, managing your environment, and doing your bidding. In many respects, Advanced Revelation works like an operating system—you issue commands that Advanced Revelation executes for you, you store data in files using its conventions, you even write programs that Advanced Revelation interprets.

Once you are in the environment, you can develop your application using all the tools available to you, including window development tools, database management tools, programming tools, etc. You have within Advanced Revelation everything you need to create complete applications.

When your application is complete, you can distribute it to other users. Because Advanced Revelation is an interpretive environment, users need their own copy of Advanced Revelation to use your windows, data tables, and programs. If users want to develop your application further, they can use a full development copy of Advanced Revelation. If they want only to run your application, but not change it, they can use a Runtime version of the Advanced Revelation environment.

Template-driven

Much of the development work that you do in Advanced Revelation produces *templates*. For example, when you create a window in Advanced Revelation, you are not really creating a program. Instead, you create a window template (definition)—an

entry in a table that contains all the information that Advanced Revelation needs in order to produce the window that you have specified.

When you want to display the window for a user to enter data, you call Advanced Revelation's window utility. The utility takes the window template you've created and dynamically creates and displays the window exactly the way you've specified it.

The same is true for the other elements of your Advanced Revelation applications. If you create a menu, you are really just creating a menu template that is interpreted at the time you display the menu. Popup windows (lists) are done the same way.

In fact, even programs in Advanced Revelation are interpreted. To write programs in Advanced Revelation you write source code, as you would in any language. You then compile the code using Advanced Revelation's compiler, which produces object code. However, you don't run the object code directly at the operating system (that is, you don't produce COM or EXE files). To execute the code, you run it *within* Advanced Revelation. Advanced Revelation's "engine" interprets and executes the object code, performing all the operations that you've written into your program.

Benefits of a template-driven system

The great benefit of a system based on templates is *flexibility*. Changing your system is as easy as calling up the template, making the change, and saving the template. The next time you run the window or menu or popup, the system reads the changed template and displays the updated version for you.

This is particularly useful when you are developing systems. You can easily prototype your system by painting a simple window and menu. As the specifications for the system develop, you can modify and extend the prototypes with simple changes to the templates.

In fact, you never need to discard the prototype at all. You can keep extending the prototype until it *is* the finished system. If you find, as many developers do, that users want to keep making changes to the system even after it has been installed, it's an easy job to modify the existing templates and update a user's system by copying the templates to the user's version of Advanced Revelation.

Let's get started ...

There are far too many features and benefits of Advanced Revelation for us to discuss them all here. Instead of listing them, we encourage you to jump right in. First, of course, you'll want to install your new copy of Advanced Revelation. If you like, you can then use the fully interactive tutorial to start your tour, or you can call up the sample application and see how applications look and feel. As with Advanced Revelation itself, the possibilities are endless ...

2. About the documentation

About the documentation

The documentation is divided into three manuals:

- *User's Guide*
- *Reference*
- *R/BASIC*

The *User's Guide* covers all aspects of creating and modifying an Advanced Revelation application. The *Reference* manual includes sections on R/LIST, Advanced Revelation's reporting language, SQL, structured query language, and TCL, Advanced Revelation's command language. The *R/BASIC* manual includes an introduction to programming with R/BASIC and a comprehensive command reference section.

Conventions used in the documentation

The following conventions are used in the documentation. Refer to the following as you read, especially if you are reading about TCL, SQL, or R/BASIC commands.

Convention	Meaning

⇨	Continuation. Text is split in the book because of margins, but should be entered on one line in Advanced Revelation. Example:
	`LIST SAMPLE CUSTOMER COMPANY NAME ADDRESS ⇨`
	`CITY BY CITY`

[ ]	a) Optional expression. The text or expression in the square brackets is an optional part of the command or expression. Example: `PAINT [tablename] template name`
|	Separates an either/or choice. Example: `WHILE | UNTIL`
...	Specifies that the preceding item may be repeated. Example: `LIST tablename [column_name ... ]`

Convention	Meaning
lower case	Variable information in an example. The word in lower case indicates a position that will be filled with a real name or expression when you actually execute a command. (See also Italics.) Example: `LIST tablename BY sort criteria`
Bold face	Menu option. Boldfaced words refer to the literal names of items on an Advanced Revelation menu. Example: Choose the **Define-Window** option ...
Italics	a) Names of prompts in windows appear in italics. Example At the *Table name* prompt ... b) Italics are used for variable information if the text around it is already in lower case. The name *tablename* is not valid ...

`Courier text`	Examples or messages. Text in courier indicates an example or a message that you might see in Advanced Revelation. Examples:
	
	`LISTDICT tablename`
	
	`Creating DICT.tablename`

Keystrokes	Keystrokes are enclosed in square brackets ([ ]). Keystrokes that are entered together are enclosed in one set of matching brackets. If a series of keystrokes are to be entered, each keystroke is enclosed in matching brackets. Examples: [Ctrl-F2] Press [Esc], then [F9].
Mouse buttons	Buttons are enclosed in angle brackets as they appear on the status line. Example: <Save>

Convention	Meaning
Caution!	Caution notes warn you of actions that might cause data loss. Example:

> **Caution!** Pressing [Alt-D] twice in the window will delete the row.

Icons

Icon	Meaning
	Instructions for mouse users
	Instructions for keyboard users

3. Installing Advanced Revelation

Installing Advanced Revelation

What you need to install Advanced Revelation

- At least 7 MB disk space on a hard disk. The disk space requirement is subject to change and may increase. The installation program will let you know if there is not enough space available on your hard disk.
- One 5.25" or one 3.5" disk drive.
- Proper read/write rights to the target directory. This is particularly important if installing directly onto a network drive. Because Advanced Revelation creates its own subdirectories, be sure that you also have subdirectory creation (parental) rights.

What you need to run Advanced Revelation

	Stand-alone application	Networked Application
RAM	640K	1 MB*
Processor	80286	80286
Operating system	DOS 3.X	DOS 3.X

* 2 MB for client-server applications

Enhancing general performance

To optimize Advanced Revelation's performance, your system should have:

- 80386/80486 Processor
- 4 MB RAM
- DOS 5.0
- VGA Color Monitor
- Mouse or other pointing device
- Expanded Memory Manager software such as 386^{MAX} software-emulated expanded memory from Qualitas, Inc. or Quarterdeck Expanded Memory Manager software-emulated expanded memory from Quarterdeck Office Systems.

Installing Advanced Revelation

During the Advanced Revelation installation process, the system determines if there is enough space on the destination directory and if there are any tables on the destination directory. If any tables are present in the destination path directory, the installation process asks what should be done with the tables:

- C (Clear tables) erases all of the tables in the destination directory prior to installing Advanced Revelation there.
- O (Overwrite tables) ignores the existing tables and copies the Advanced Revelation tables to the destination directory. An existing table with the same name as an Advanced Revelation table will be overwritten.
- A (Abort) cancels the installation process and returns to the operating system.

To begin the install:

1. Insert Advanced Revelation System Disk 1 into disk drive A or B and enter the drive letter.
2. At the operating system prompt, enter:

   ```
   INSTALL
   ```

3. The installation program asks you to verify:
   - The source path (the drive from which you are installing Advanced Revelation.)
   - The destination path (the target drive and the complete directory path where Advanced Revelation tables will be installed.)

   Change either one of the paths if appropriate. Press [F9] to save your changes.
4. Press [Enter] to start the installation.
5. Follow the instructions displayed on the screen to complete the installation.

Starting Advanced Revelation

To start Advanced Revelation:

- Move to the directory that contains Advanced Revelation.
- Type AREV.

Note You must be in the subdirectory where Advanced Revelation resides. Do not use a PATH or MAP command to get to Advanced Revelation.

Logging on as a user or an application

For more information about applications and users, see the "Creating and managing applications and users" in the *User's Guide*,

To log onto a specific application or as a specific user, enter the name after AREV. To log directly into the TUTOR application enter:

```
AREV TUTOR
```

Entering start up options

To enter start up options, enter a forward slash (/) and the logon options after AREV. For example, to suppress the logon banner, at the operating system prompt enter:

```
AREV /S
```

To use more than one logon option when you start Advanced Revelation, enter a forward slash (/) and then the logon options separated by a space. For example, to suppress the logon banner and use expanded memory, at the operating system prompt enter:

```
AREV /S X
```

A full list of logon options appears in Appendix 4 of the *Reference* book.

Accessing Advanced Revelation from Microsoft Windows

A PIF (program information file) and an icon for Advanced Revelation can be found in the Advanced Revelation subdirectory. To add these files to Microsoft Windows, refer to the MS Windows documentation on creating groups and adding items to groups.

Accessing Advanced Revelation from OS/2 2.0

You can create an Advanced Revelation program object on your OS/2 desktop with the Migrate Applications program or by copying a new program object from the Templates folder. An icon for your desktop, AREVOS2.ICO, can be found in the Advanced Revelation subdirectory. Refer to the OS/2 documentation for more information on creating program objects and customizing program settings.

Leaving Advanced Revelation

Quitting Advanced Revelation

To end an Advanced Revelation session:

- Choose the **Application-Quit** option from the Opening menu.
- Choose the **Exit-Quit** option from the Development menu.
- Enter the command OFF at the Command window.

Closing an application

You can close the application in which you are working and return to the Opening menu by selecting the **Exit-Close** option from the Development menu.

Exiting Advanced Revelation temporarily

You can exit Advanced Revelation temporarily if you need to run one or more operating system commands. To exit temporarily

- Choose the **Application-DOS** option from the Opening menu.
- Choose the **Exit-DOS** option from the Development menu.
- Enter the command SUSPEND at the Command window.

For more information, see "TCL Command reference " in *Reference.*

SUSPEND saves most of Advanced Revelation to a temporary disk file. Most of the RAM used by Advanced Revelation will be released. You can use almost any command or program at the operating system level.

To return to Advanced Revelation, type EXIT at the operating system prompt.

Caution! Do not run any memory-resident utilities while Advanced Revelation is suspended.

4. Orienting yourself in Advanced Revelation

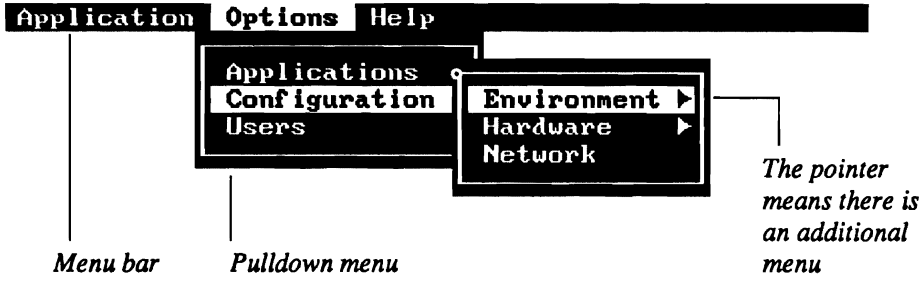
About the Advanced Revelation environment

Advanced Revelation presents you with an integrated environment for developing and using applications. To make best use of Advanced Revelation, you will want to become familiar with these major components:

- menus
- status line
- windows
- popups
- messages
- the Command window (TCL), Advanced Revelation's command-line interface
- help

Menus

You use menus in Advanced Revelation to navigate around the system. Selecting a menu option can display a window, start a process, or display another menu.

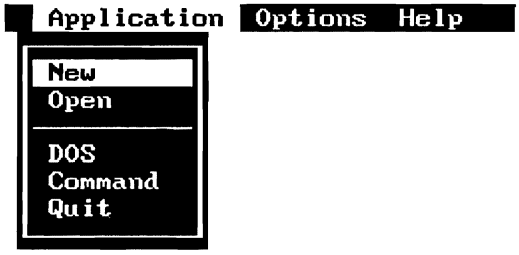


Anchored menus

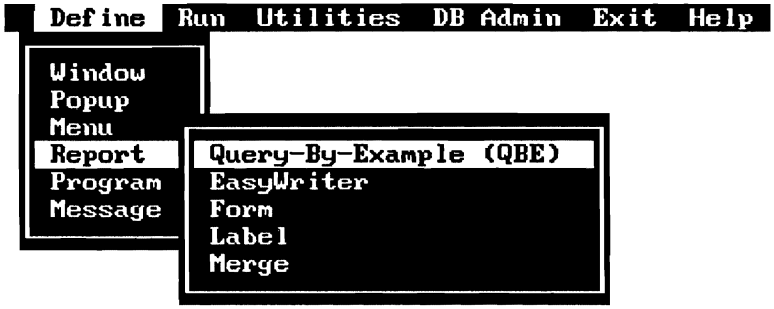
A menu can be anchored or non-anchored. If a menu is anchored, the window or process it calls will return back to the menu when done. If a menu is non-anchored, it will call a window or process, but when the process is done it returns to the last anchored menu rather than the calling menu.

System menus

When you first log onto Advanced Revelation, the Opening menu displays at the top of the screen. From the Opening menu, you can create new applications and users, open existing applications, and change the configuration of your system.



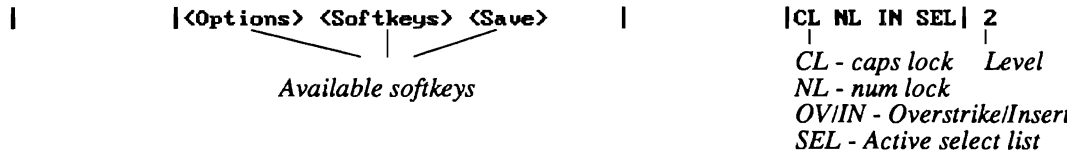
Once you have created a new application and opened it, the Development menu displays at the top of the screen. From the Development menu, you can create windows, menus, programs and all other parts of your application.



The status line

The status line at the bottom of the screen provides you with information about:

- Buttons you can click to execute a process.
- Status of a process you have initiated.
- Status of your [NumLock] or [CapLock] keys.
- The level you are at within Advanced Revelation.



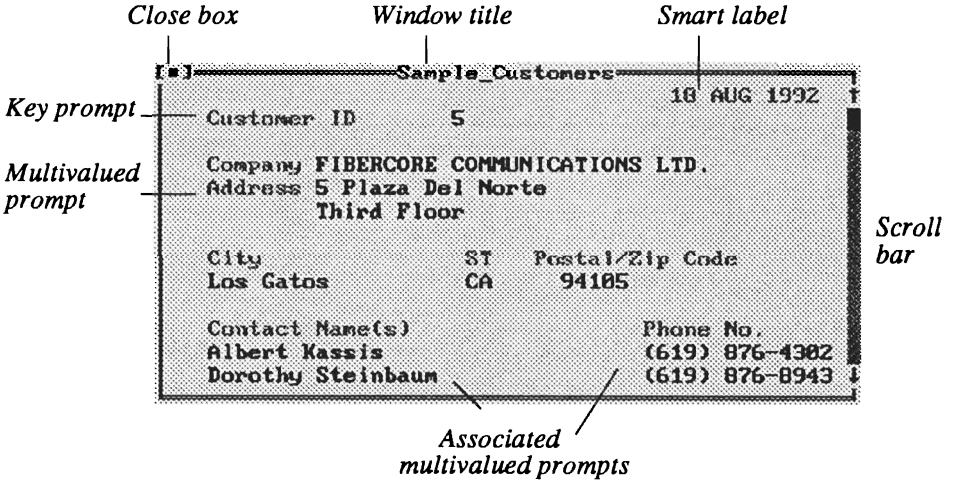
To activate a button in the status line, place the pointer on the button and click either mouse button.

Turning off the status line

Display of the status line is optional. You can turn off status line for a user or for an application as a whole. For more information, see the chapter "Creating and managing applications and users".

Windows

Windows are bordered areas on the screen that contain prompts and entry lines.



Prompts and labels

Types of prompts

- Single value prompts allow you to enter a a single value on one line.
- Multivalued prompts allow you to enter a series of distinct values.
- Associated multivalued prompts are a group of related multivalued prompts.
- Text prompts accept a string of text that extends over more than one line.

Key prompts

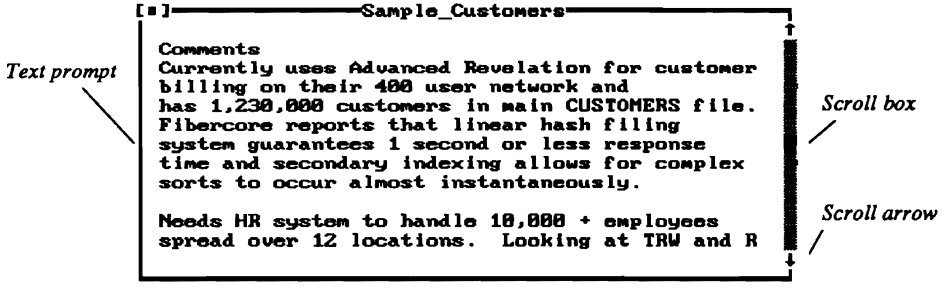
In Advanced Revelation, every row is uniquely identified by a key. In a data window, the cursor always starts at the key prompt. Until the key prompt is filled in, you cannot move to another prompt.

Labels

Labels are text or characters that are displayed in the window but not associated with any prompt or entry line.

Scroll bars

A window can be larger than the visual display area - up to 160 columns by 200 rows. Scroll bars on the right border or the bottom border indicate that there are additional sections of the window to view.



Types of windows

- Application windows are used to enter, modify and view data into tables. Information entered in an application window is stored in a data table.

- Collector windows gather information for a process you are executing. The information entered in a collector window is not stored in a data table.

Softkeys

Softkeys are developer-defined keystrokes that are tied to a specific window. The keys available for softkeys are [Shift-F1] through [Shift-F10] and [Alt-F1] through [Alt-F10].

If softkeys are available, a softkey button appears in the status line. You can also display available softkeys by pressing [F6].

Related windows

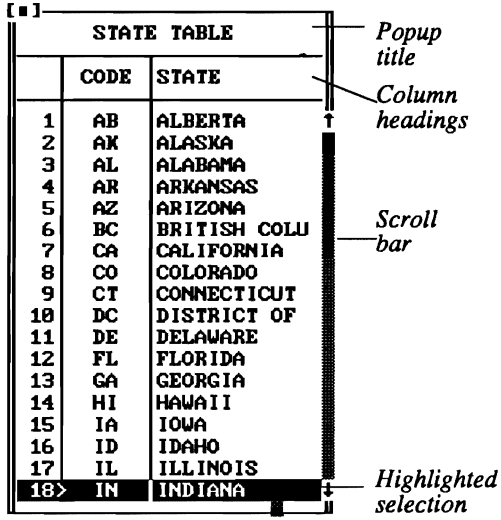
Several windows can be displayed at a time based on data they have in common. With related windows, you do not need to leave the current window to look up or modify information in another window. To display available related windows, press [Ctrl-F6] in the window.

Popups

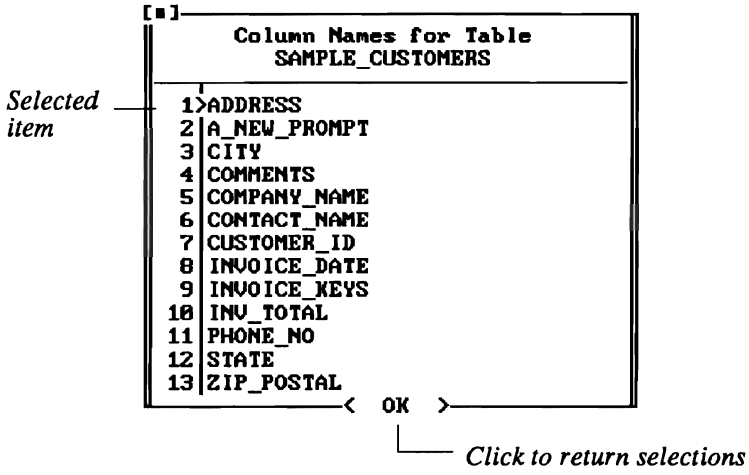
Popups display information or list options. A popup will return one or more selected values to the location from which it was called, perform an action such as a softkey action, or show you a list (read-only).

Types of popups

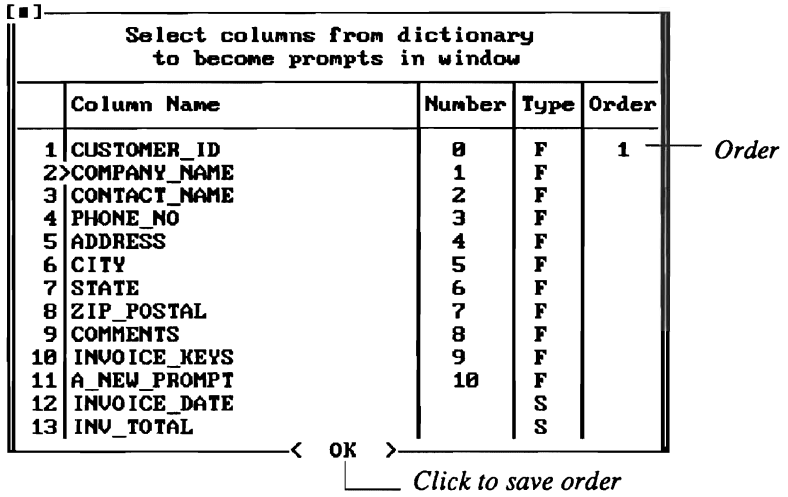
- Single-selection popups allow you to chose a single option and return.



- Multiple-selection popups allow you to choose one or more options and return.



- Sort popups allow you to reorder elements.



- Toggle popups allow you to alternate or toggle between two choices.
- Display-only popups display system information or help messages.

Interpreting the status line

When a popup appears, the status line displays the type of popup in the first cell.

```
Order  |Select row [Return] in the order of choice [F9]-Save    |  NL      | 4
```

Type of popup *Selection instructions*

Cell 1	Popup Type
Single	Single selection
Multi	Multiple-selection
Display	Display only
Order	Sort

The second cell in the status line displays a message that explains how to make selections from the current popup.

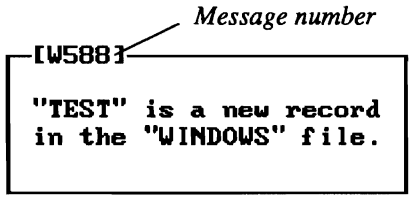
Messages

For more information, refer to "Programming with R/BASIC" in *R/BASIC*.

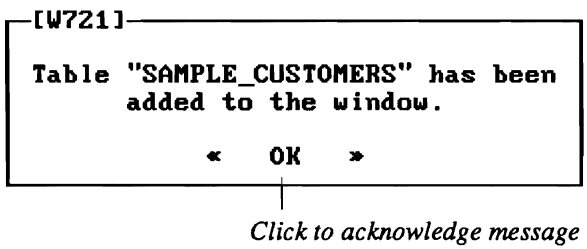
Messages provide or ask for information, such as reporting errors or requesting information for a process.

Types of messages

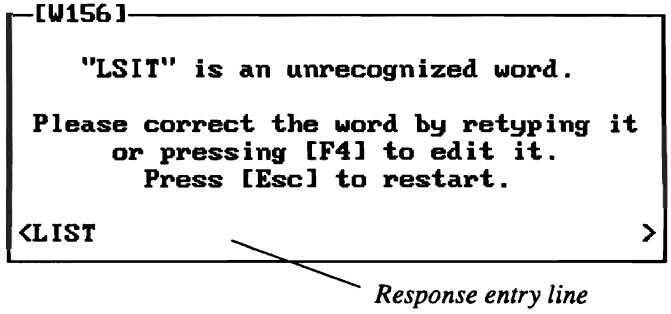
- Timed messages appear for a specified period of time and provide information to you.



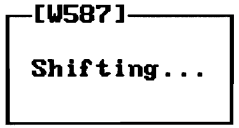
- Acknowledged messages display until a key is pressed or the <OK> button is clicked.



- Response messages ask for information or user input.



- Progress messages report the status of the current process.



Messages display the message number in the upper border of the message.

The Command window (TCL)

For more information, refer to the "TCL command reference" in the *+Reference* manual.

TCL is Advanced Revelation's interactive command language. You can use TCL to:

- Navigate through Advanced Revelation.
- Move directly to a window or other system tool.
- Start a process with a number of options already set.

Enter TCL commands at the Command window.

Displaying the Command window

- Press [F5] to display the Command window.
- Choose the **Application-Command** option from the Opening menu.
- Choose the **Exit-Command** option from the Development menu.

Getting online help

For more information, see "Building applications" in the *User's Guide*.

Online help is available throughout Advanced Revelation.

Types of help

When you want to	Press this key	To display
Read help text about the current prompt or popup	[F1]	Context
Read help text about the concept behind the current window or process	[Ctrl-F2]	Concept

(continued)

When you want to	Press this key	To display
Choose help for a window, command, or function from a menu	[Ctrl-F1]	General
View a list of all keys that are currently active	[Ctrl-F9]	Available keys
Display a list of options	[F2]	Options
Display a list of available softkeys	[F6]	Softkeys

Using the online tutorial

Advanced Revelation's online tutorial is the perfect way to learn how to use Advanced Revelation. Once you get started, almost everything you need to know is right there on the screen for you.

There are over a dozen lessons on major aspects of Advanced Revelation. As you work with the tutorial, you actually create windows, menus and popups.

Each lesson stands alone. It is not necessary to complete them in any particular order.

Starting the online tutorial

To start the tutorial:

- Choose the **Help-Tutorial** option from the Opening menu or the Development menu.
- Enter the TCL command TUTORIAL at the Command window.

Note Only one user can run the tutorial at a time. If you try to access the tutorial while another person is running it, a message displays asking that you try to run the tutorial at a later time.

Starting a lesson

To start a lesson, select the lesson you want to run from the popup of available lessons. From the Opening menu, there are two lessons available. From the Development menu, there are more than a dozen lessons available.

Active keys in the tutorial

When you want to	Press
Return to the tutorial lessons popup or quit the tutorial	[Esc]
Display the Command window	[F5]
Speed up demonstration keystroke playback	[+]
Slow down demonstration keystroke playback	[-]
Clear a Tutorial message	[Enter]

Stopping a lesson

You can interrupt any lesson by pressing [Esc]. When you press [Esc] a message like this one will display:

```
Leave the tutorial lesson?
Are you sure?
<<Yes>> <No>
```

Enter "Y" to exit to the tutorial lesson popup.

Leaving the tutorial temporarily

To suspend a tutorial lesson and pursue another task, press [F5] to display the Command window or press [F10] to enter the menu system. The tutorial displays a message similar to this one:

```
Temporarily leave the tutorial?
Are you sure?
<<Yes>> <No>
```

Enter "Y" to confirm that you want to leave.

When you are ready to return to the suspended menu, press [Esc] to return to the tutorial lesson.

When you leave the tutorial temporarily, Advanced Revelation continues to view you as a tutorial user. Tutorial tables will be substituted for some of the tables that are normally available to you. for example, when you use Paint, you will access the tutorial's WINDOWS table.

Note To preserve the integrity of the tutorial, do not make any changes to existing rows in tutorial tables. Doing so may cause unpredictable results when the tutorial is run.

Quitting the tutorial

To quit the tutorial, press [Esc] to return to the tutorial lesson popup, then press [Esc] again.

5. Navigating through Advanced Revelation

Using menus

Use menus to navigate throughout Advanced Revelation.

Selecting menu options

To select a menu option:

- Click the menu selection you want.

- Type the highlighted letter of the menu selection.

Active keys in menus

If you want to	Press
Activate (or display) a menu	[F10]
Display a list of all available menus	[F2]
Exit to last menu	[Esc]
Select current menu item	[Enter]
Move cursor forward one option	[↓] or [→] or [Space]
Move cursor back one option	[↑] or [←] or [Backspace]
Move cursor to first option	[Home]
Move cursor to last option	[End]

Using windows

Use windows to issue commands, design an application or modify data. In data windows, you cannot move to other prompts until the key is entered.

Moving from page to page

If a window is larger than the displayed area on the screen, the application developer may have added page tabs to the window when it was designed. When page tabs are set, you can move from page to page (page tab to page tab) in a window. To do so, press [PgUp] and [PgDn].

If a window is larger than the visual display area, there will be scroll bars on the right border, the bottom border or both borders.

Moving from page to page operates in a circular fashion. You can move consecutively through the pages, going forward or back. If you page down to the last page, you circle back to the beginning.

Editing and entering data

Display length

If you enter data longer than the display length, the data scrolls to the left. Once you start to move to the next prompt, all of the characters may not display. However, all of the data will be saved.

Protected prompts

Application developers may protect the data displayed at some of the prompts in a window. There are three types of protected prompts:

Type	Cursor Action
Protected	Cursor will not stop at prompt
Filled and protected	Window will automatically fill in default data as the cursor skips over it.
Required and protected	Window will prompt for data in new rows and protect the data from then on.

Resetting a sequential counter

1. At the key prompt, type [=].
2. Enter the new default value in the window that displays.

Edit mode

For more information, see "Configuring the workstation" in the *User's Guide*.

By default, when you begin entering data at a a prompt, any existing data will be cleared first. You can change your edit mode to non-destructive (data is inserted in front of existing data) in the Editor Environment window. You can temporarily change your edit mode by pressing [Ins].

To make a permanent change in your editing environment, from the Opening menu choose **Options-Configuration-Editor**. In the Editor Environment window, specify "No" at the *Destructive Edit Toggle* prompt.

Expanding the area for a multi-lined prompt

To create a larger editing area for a multi-lined prompt, press the Zoom key [F3]. The editing area for the current prompt will expand to fill one half of the screen.

Changing settings when editing text prompts

Press [Ctrl-W] to change some of the settings in the Editor. The settings which you can alter are:

Setting	Action
Insert mode	If On, characters are inserted within existing text rather than overwriting them.
Wrap line	If On, as the data is entered, lines which are too wide are split and continued on the next line.
Inset line	If On, each time you press [Enter], a blank line is inserted.
Indent line	If On, new lines will match the indentation of the previous line.

Browse lists

While you are in a window, you can have list of keys to rows in that data table available to you. This list is called a browse list. If there is an active browse list, SEL appears in the fourth cell of the status line.

Panning through a window

Windows can be larger than the visual display area. If they are, scroll bars will appear on a window's right or bottom border, or both. Use Pan, [F7], to quickly page through a window.

Navigating in a window

Click the entry line you want.

General navigation keys

Move cursor up one prompt	[↑]
Move cursor down one prompt	[↓] or [Enter]
Move cursor left one column	[←]
Move cursor left one column (destructive)	[Backspace]
Move cursor right one column	[→]
Move cursor right one column (destructive)	[Space]
Move cursor right to next tab stop in prompt	[Tab]
Move cursor left to next tab stop in prompt	[Shift-Tab]
Move to next page if multi-page window	[PgDn]
Move to previous page if multi-page window	[PgUp]
Move cursor to first character in prompt	[Home]
Move cursor to last character in prompt	[End]
Move to key prompt	[Alt-K]
Move forward one tab prompt	[Alt-T]
Move backward one tab prompt	[Alt-Y]
Move to middle prompt	[Alt-W]
Move to selected prompt from popup	[Alt-A]
Move right one word in entry line	[Ctrl →]
Move left one word in entry line	[Ctrl ←]

General editing keys

Toggle Edit mode on and off. Undo current line changes when at edited line.	[F4]
Toggle insert/overwrite edit mode	[Ins]
Delete character at current cursor position	[Del]
Delete character to the left of the cursor	[Backspace]
Clear current row from display	[F8]
Delete current row	[Alt-D]
Copy data from previous row	[Alt-C]
Copy current entry line from previous row	[Alt-O]
Delete characters to end of line	[Ctrl-L]
Clear all characters at current line	[Ctrl-X]
Delete characters from beginning to current position	[Ctrl-K]
Delete word right	[Ctrl-Y]
Duplicate last character	[Ctrl-P]
Set tabs	[Ctrl-T]
Edit value window	[Ctrl-E]

Active edit keys for multivalued and text prompts

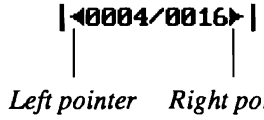
Cut line into two lines at current cursor position	[Ctrl -C]
Join current line with following line	[Ctrl-J]
Insert blank line at current position	[Ctrl-N]
Delete line at current position	[Ctrl-D]

Active keys for associated multivalued prompts

Move up one row, staying in the same column	[↑]
Move down one row, staying in the same column	[↓]
Move to the next prompt in a row, or from the last prompt in a row to the first column in the next row	[Enter]
Move one prompt left in a row of associated multivalued prompts or move to the last prompt in the previous row (editor off)	[←]
Move one prompt to the right in a row or move to the first prompt in the next row (editor off)	[Enter] or [→]
Move to prompt before associated multivalued prompts	[Ctrl-PgUp]
Move to prompt after associated multivalued prompts	[Ctrl-PgDn]
Exit the prompts	[Enter] or [↓] at the first empty row

Viewing rows in a browse list

Using the status line to review an active browse list



- To move forward in a browse list, click the right pointer in the status line.
- To move backward in a browse list, click the left pointer in the status line.
- To display the Browse Options popup, click the slash (/) in the status line.

Active keys in a browse list

Display Filter Selections popup	[Ctrl-F10]
Edit browse list	[Alt-I]
Move one row forward	[Alt-F]
Move one row backward	[Alt-B]
Print rows in browse list using template	[Alt-P]
Display list of rows changed at this window	[Alt-U]

Panning through windows

Using scroll bars

If you want to	Do this
Scroll one row	Click one of the scroll arrows
Scroll one page	Click before/above or after/beneath the scroll box
To any prompt	Drag the scroll box to a position in the scroll bar corresponding to the prompt you want to view

Active keys

Toggle Pan mode on and off	[F7]
Complete Pan	[Enter]
Pan window depth down or up	[PgDn] or [PgUp]
Pan one page to the left/right	[Home]/[End]
Pan to upper left corner of window	[Ctrl-Home]
Pan to upper left/lower right corner of window	[Ctrl-PgUp/PgDn]
Pan to lower left corner of window	[Ctrl-End]

Moving a window

Point to a window border and drag the window to its new position.

Active keys

Toggle Move mode on and off	[Ctrl-F8]
Complete Move	[Enter]
Move window up/down one row or column	[↑] / [↓]
Move window left/right one column	[←] / [→]
Move window flush to left edge of monitor	[Ctrl-Home]
Move window flush to top edge of monitor	[Ctrl-PgUp]
Move window flush to bottom of monitor	[Ctrl-PgDn]
Move window flush to right edge of monitor	[Ctrl-End]

Resizing a window

1. Point to the border or corner of the window you want to resize.
2. Drag the window border or corner until the window is the size you want it.

Active keys

Toggle Resize mode on and off	[Ctrl-F7]
Complete Resize	[Enter]
Shrink/expand window one row	[↑] / [↓]
Shrink/expand window one column	[←] / [→]
Expand/shrink one half window depth	[PgDn] / [PgUp]
Expand/shrink one half window width	[Ctrl ←] / [Ctrl →]
Expand to maximum size	[Ctrl-Home]

Miscellaneous window keys

If you want to	Press
Toggle table browser on and off	[Ctrl-F5]
Display related application windows	[Ctrl-F6]
Display available softkeys	[F6]

Closing a window

Double-click the close box in the upper-left hand corner of the window.

Press [Esc].

Using popups

Displaying a popup

Generally, when a popup is available, the status line displays:

```
<OPTIONS>
```

1. Click on the Options button.

1. Press [F2].

Highlighting options by alphabetic character

1. Type the alpha character that begins the text.
2. Press [Enter], or add text and then press [Enter].

Or:

1. Press [Ctrl-F].
2. At the *Text to find* prompt, enter the appropriate alpha characters.

Choosing items from single-selection popups

Click the item you want

1. Use the arrow keys to move the highlight to the item you want.
2. Press [F9] or [Enter].

Or:

1. Type the number of the item.
2. Press [Enter].

Choosing items from multiple-selection popups

1. Click the items you want.
2. When you have completed your selection, click the <OK> button.

1. Use the arrow keys to move the highlight to the item you want.
2. Press [Enter].
3. Repeat the first two steps to choose additional items.
4. When you have completed your selection, press [F9].

Choosing items from multiple-selection, ordered popups

Follow the instructions for selecting items from multiple-selection popups. Make sure the items are selected in the order you specify. Note that a column appears with the order in which you select items displayed.

Moving a popup

Point to a popup border and drag the popup to its new position.

If you want to	Press
Toggle Move mode on and off	[Ctrl-F8]
Complete Move	[Enter]
Move popup up/down one row	[↑] / [↓]
Move popup left/right one column	[←] / [→]

Responding to messages

For messages with buttons, for example <OK> , click the button you want.

- For messages that display one response, press [Enter].
- For messages that display more than one response option, type the first character of the response and press [Enter].
- For messages that ask for a response, type the response and press [Enter].

6. Updating and enhancing your Advanced Revelation system

Installing a maintenance update or product upgrade

Follow these instructions to install Product Upgrades and Maintenance Updates for your installed copy of Advanced Revelation 3.0 or higher.

Caution! Before running the any Advanced Revelation upgrade or update, check the CHANGES.DOC and README.DOC files on the Upgrade disk. ***These files contain information specific to the upgrade you are installing, including information about what changes the upgrade will make to your system.*** Note also that existing Environmental Bonds may not be compatible with Advanced Revelation 3.0.

The Advanced Revelation Upgrade copies information from the Product Upgrade or Maintenance Update disks directly into your base set of Advanced Revelation system tables on the SYSPROG application. Once the process finishes, you are logged out of Advanced Revelation. When you log back in, the new version of Advanced Revelation is available for use in the SYSPROG application.

If you are installing this software on a network, you must have the ability to create new files and subdirectories and to read from and write to them.

Caution! Be sure to back up your system before and after installing an Update or Upgrade.

These instructions apply to Advanced Revelation for DOS only. If you are using Advanced Revelation for OS/2, special instructions may be provided for your Product Upgrade or Maintenance Update. In this case, follow the procedure supplied with your new disks.

Upgrading optional modules

If you have optional modules from Revelation Technologies installed on your copy of Advanced Revelation (for example, an add-on Environmental Bond), a product upgrade will upgrade these as well—*unless the upgrade documentation specifically states otherwise.* Optional modules from other vendors might not be compatible with version 3.0 of Advanced Revelation, so be sure to check with your vendor before upgrading your system.

If you have an optional module that you want to use but have not yet installed, generally speaking you should install it *before* you attempt to upgrade your system (but not if the upgrade is incompatible with that optional module). The upgrade process will then upgrade your optional module as well. If you attempt to install an old optional module onto a system that has been upgraded, it might not work, especially if the optional module is version-specific.

Note Always check the documentation (including all README and CHANGES files) before upgrading to determine what the effect of the upgrade will be on any optional modules that you are using.

Update and upgrade process options

When you install a maintenance update or product upgrade, you have three options:

- REPORT lists tables and rows that will be affected by the installation. This report will help you decide if the installation process will interfere with customizations that you have made to your Advanced Revelation system.
- TEST checks all of the disks, one by one, for errors. In addition, TEST warns you if any system tables required by the upgrade are not attached. This process takes several minutes.

  Caution! If one or more system tables are not attached when you install an update or upgrade, your system will not be fully upgraded and may not work properly.

- UPGRADE starts the installation process. Once you have selected UPGRADE, you cannot stop or reverse the process.

  Note Check the Update or Upgrade documentation for any special installation instructions. In addition, check for CHANGES.DOC and README.DOC files on the Update or Upgrade disk for compatibility information.

Installation steps

For more information, see "Note to the system administrator" in the "Appendix"

To install a maintenance update or product upgrade, follow these steps:

1. Back up your Advanced Revelation system tables. The installation process adds and overwrites rows in most of these tables but leaves your application tables intact.

   Caution! The installation process makes permanent, irreversible changes to your copy of Advanced Revelation. If installation aborts or fails, you must restore your system from a backup copy.

2. If you are using a network version of Advanced Revelation, make sure that other stations log off of Advanced Revelation before you begin the installation.
3. Log on to your existing copy of Advanced Revelation as the INSTALL user. At the operating system prompt, enter:

   ```
   AREV INSTALL
   ```

4. Select the **Install** option at the Installation Utilities menu.
5. Insert disk 1 of your disk set and enter the name of the source drive.
6. The Installation program displays a popup window with three options. Run the installation options in order (REPORT, TEST, and UPGRADE), and follow the screen instructions, or press [Esc] to cancel the installation process.

Caution! Once you have begun installation, do not attempt to cancel it before completion. If you attempt to cancel in mid-installation, the module will not be fully installed, and your system may not work properly.

7. If you have optional modules to install, you can do so now.

For more information, see "Installing optional modules" in this chapter.

After installing an Update or Upgrade, you *must* run the Upgrade Applications process to make the new Upgrade or Update available to users in other applications on the system.

Upgrading applications

Advanced Revelation's Upgrade process updates a base set of system tables in the SYSPROG application. The base set of tables consists of the Advanced Revelation system tables that are attached at the time that the new software is installed.

However, each application you create also has its own set of system tables. For example, when you create a new application, that application has its own copy of the VOC table. Moreover, you might have decided to make your own copy of one of Advanced Revelation's own system tables, such as the SYSHELP table, the SYSMESSAGES table, or another.

The Upgrade Applications process is designed to upgrade these sets of tables so that they match the base set of updated tables.

You must run the Upgrade Application process for each application after you install optional modules, an Advanced Revelation Product Upgrade, or a Maintenance Update. The Upgrade Application process updates tables in each of these applications, making the changes available to users in those applications. You can either:

- Log on to the SYSPROG application and update tables in all (or selected) user applications.
- Log on to a user application and update the currently attached tables.

The Upgrade Application process uses the base set of tables on the SYSPROG application as the basis for updating tables in other applications.

Note Be sure to back up your system after upgrading user applications.

Upgrading applications from SYSPROG

If you upgrade applications from the SYSPROG application, you have four options:

ALL APPLICATIONS

This option updates all non-SYSPROG applications in one pass. ALL APPLICATIONS identifies all current applications, allows you to select the applications to upgrade, and then makes all necessary changes to application system tables.

When you select ALL APPLICATIONS, a list of the tables to be updated is displayed. For each application, you are prompted to enter the location or volume name of the current version of each table (the default is REVBOOT, the location where you have installed Advanced Revelation). This enables you to update the appropriate tables for each application regardless of where the tables are stored.

ATTACHED TABLES

The ATTACHED TABLES option automatically updates all of the application system tables that are currently attached. You would use this option if you had opened up the application to update and had already attached all the tables for that application. The base set of tables on the SYSPROG application is used as the basis for updating local versions of these tables.

When you select the ATTACHED TABLES option, a list of the tables to be updated displays. Enter "Y " at the prompt to confirm that you want to update the attached tables. If a table has been updated already, it will not be updated again.

SINGLE TABLE

The SINGLE TABLE option updates the version of a system table that is currently attached. For example, if you have a copy of the SYSHELP table that was not accessible to the Installation process, you can update it using this option. The base set of tables on the SYSPROG application are used as the basis for updating the local version of a table.

SINGLE TABLE does not update the dictionary of a table automatically. To update a table and its dictionary, you must run SINGLE TABLE twice, once for the table and once for the dictionary. If a system table has been updated already, it will not be updated again.

USER TABLES

The USER TABLES option displays a menu with four selections:

- Choose **Tables** to update table and column names to conform to naming conventions implemented in Advanced Revelation 3.0.

  Caution! Do not run the **Tables** option more than once. See the README.DOC or CHANGES.DOC files for additional information about changes to table names.

- Choose **Users** to update user entries in system tables and change user names to conform to current naming conventions.
- Choose **Reports** to copy report definitions to the one system table, REPORTS, which stores all report definitions.
- Choose **Tools** to change the location of application windows, popups and messages to allow for the display of menus.

Executing the upgrade application process in SYSPROG

To execute the Upgrade Application process, follow these steps:

1. Make sure that no users are logged on to applications that you intend to update. Other users should log out while the update is executing.
2. Log on to Advanced Revelation without any user or application name so that you open the SYSPROG application
3. From the Opening menu, choose the **Options-Application** option.
4. At the Application window, press [F10] to activate the Application menu and select the **Upgrade** option.
5. The Upgrade program displays a popup window listing the four options described above: ALL APPLICATIONS, ATTACHED TABLES, SINGLE TABLE and USER TABLES. Choose an option and follow the screen instructions, or press [Esc] to return to the menu.
6. After determining that the correct user applications have been updated, you can delete the UPDATE.LOG and UPDATE.CTL tables that are in the subdirectory containing the AREV.EXE file.

   Note Upgrade Application is also an option on the Installation Utilities menu. Enter AREV INSTALL at the operating system prompt to access this menu.

For more information, see "Note to the system administrator" in the Appendix to the *Reference* manual.

Upgrading user applications from a non-SYSPROG application

Note If there is a password on the SYSPROG application, you cannot run the Upgrade Application process from a non-SYSPROG application. A message will report that the process cannot attach a table in the SYSPROG application. When the message clears, you are returned to the Application Management menu.

To execute the Upgrade Application process from a non-SYSPROG application, follow these steps:

1. Log on to the application as you normally do.
2. Press [F5] to display the Command window.
3. Enter:

   ```
   CW ACCOUNT
   ```

4. Enter the name of the application you want to upgrade at the *Name* prompt in the Application window.
5. Press [F10] and choose the **Upgrade** option from the menu.
6. The Upgrade program displays a popup window listing the four options described above: ALL APPLICATIONS, ATTACHED TABLES, SINGLE TABLE and USER TABLES. Choose an option and follow the screen instructions, or press [Esc] to return to the menu.

Installing optional modules

Follow these instructions to add optional modules, including Environmental Bonds, third-party software, and system utilities, to a copy of Advanced Revelation 3.0 or higher.

Caution! Due to changes in system table names and system commands, some optional modules may not be compatible with Advanced Revelation 3.0. Check with your optional module vendor before installing the module.

Note The Advanced Revelation Installation Utility can install many, but not all, third-party software products. Refer to the documentation provided with the software (or provided by your developer) for installation instructions.

When installation is complete, you are logged off of Advanced Revelation. When you log back on, the newly installed modules are available for use in the SYSPROG

application. From then on, whenever you create a new application, all installed optional modules are made available to the new application automatically.

Adding optional modules to an existing system

For more information, see "Installing a maintenance update or product upgrade" and "Upgrading user applications" in this chapter.

Before you add optional modules to an existing system, be sure your copy of Advanced Revelation is at the proper release level to support the module you want to install. For example, if the optional module you want to install requires version 3.0 of Advanced Revelation, you must upgrade an older version of Advanced Revelation before you can install the optional module.

After installing new modules, you must run the Upgrade Application process to make the modules available to users in other applications on the system.

Optional module installation options

Once you start the installation process, you have three options:

- REPORT lists tables and rows that will be affected by the installation. This report will help you decide if the installation process will interfere with customizations that you have made to your Advanced Revelation system.
- TEST checks all of the disks, one by one, for errors. This process takes several minutes.
- INSTALL starts the installation process. Once you have selected INSTALL, you cannot stop or reverse the process.

Installation steps

To install an option module, follow these steps:

1. Complete steps 1 through 5 for the update and upgrade installation process described earlier in this chapter.
2. Advanced Revelation displays a popup that lists the modules provided on this disk set. Highlight the modules you want to install by pressing [Enter] or clicking on the modules you want to install.
3. Press [F9] or click <OK> to begin installation.
4. Run the installation options in order (REPORT, TEST, and INSTALL), and follow the screen instructions, or press [Esc] to cancel the installation process.

Note If you find that the Installation program cannot read the first disk in the optional module disk set, it is likely that this third-party product must be installed some other way. Refer to the documentation supplied with your disks.

5. When the installation is complete, you will be logged out
6. If you are adding optional modules to an existing system, you now must run the Upgrade Application process to make the new modules available to users in the existing applications on the system.

For more information, see "Upgrading user applications" in this chapter.

Optional modules currently available

In Advanced Revelation, the WHO window lists the optional modules that are currently available on your Advanced Revelation system. To display the optional module information:

1. Log in to Advanced Revelation.
2. Choose the **Help-System Info** option from the Opening or Development menus.
3. When the WHO window displays, press [PgDn] to view optional module information.

Removing optional modules

When you run the Remove process, it will remove the module and the routines that support it from your base set of Advanced Revelation system tables on the SYSPROG application. The "base set" of tables consists of the Advanced Revelation system tables that are currently attached, including all global tables such as SYSWINDOWS, SYSPOPUPS, SYSMENUS, etc.

Note The Remove process only removes optional modules that were installed using the Advanced Revelation Installation Utility programs. The Installation process is described in the "Installing Advanced Revelation" chapter.

To remove an optional module, follow these steps:

1. If you are using a network version of Advanced Revelation, make sure that only your station is using Advanced Revelation while the Remove process is executing. Other stations should log out of Advanced Revelation before you begin the Remove process.
2. Log on to your existing copy of Advanced Revelation as the INSTALL user. At the operating system prompt, enter:

   ```
   AREV INSTALL
   ```

3. From the Installation Utilities menu, choose the **Remove** option.

4. Advanced Revelation displays a popup that lists the optional module(s) that are currently available on your Advanced Revelation system. Select the module(s) that you want to remove and press [F9] or click <OK> to begin the removal process.
5. Advanced Revelation will identify the module that it is currently removing. Another message displays when the removal is complete. You are automatically logged out of Advanced Revelation.

Building applications

7. Creating applications and users

About applications

An application in Advanced Revelation is an integrated set of tables and processes that enable users to do a particular job. This might be everything from a small mailing-label database to a full suite of accounting modules.

What's in an application?

When you create an application, what do you end up with? Stated simply, the end result is a group of tables that contain the results of your development efforts. A typical application consists of:

- Tables for the data required for your application. These might have data in them (for example, a table of all the zip codes in the US), or might be empty, waiting for users to enter data into them.
- Dictionaries that contain the column definitions for the data tables.
- Indexes, if you want to make it faster to search and sort the data in the application.
- One or more tables that contain the definitions of the windows you've created to help users enter data into the tables.
- Tables that contain the popup, menu, and message definitions you've created to guide the user through the application.
- Tables that contain report definitions to display data in various meaningful ways.
- Tables that contain R/BASIC programs that manage and manipulate data, such as calculating values and printing invoices.

An unusual feature of Advanced Revelation that you can see from this list is that *everything* you do results in an entry in a table. All the data is in tables, of course. But all the development you do also results in entries in a table. When you create a window, its definition is stored as one entry in a table. The same is true for popups, menus, and messages. Even R/BASIC programs are nothing more than entries in a table—both the source code and the object code. The finished application, as noted, is the group of tables that contain all these elements.

Why set up applications?

As a developer, you use Advanced Revelation's development tools to create the tables and processes that the users need. The end result is a finished application. You then provide this finished application to the users, who often don't know anything about Advanced Revelation or about the development tools in it—the only thing they ever see is their own application.

The advantages of separating your work into different application are:

- **Security**. People working in one application can't access information in another application unless you allow them to.
- **Maintainability**. If you segregate applications (and especially if you keep them in different locations, as recommended), they are easier to move, copy, and back up.
- **Ease of use**. If you segregate applications, you only have to see and work with one set of tables, windows, commands, etc., rather than having to sort through the elements of many different projects in one place.

You can create any number of applications using your copy of Advanced Revelation. Each will maintain its own set of tables and characteristics, so when you are in an application, the only tables you have available are those that belong with the current application (or those that are global).

The SYSPROG application

When you log onto Advanced Revelation and see the Opening menu, you are in a special application called SYSPROG (system programmer). SYSPROG is the application from which all other applications are created. SYSPROG is also where you define new users and change application and user characteristics.

The scope of an application: when is an application an application?

When we say "application", what do we mean? Within Advanced Revelation we have a clear, technical definition: an application is a group of tables that are all have the name of the application stamped on them. Stated another way, they are tagged in such a way that the application "owns" them. As an extension, we consider an application to be a group of tables, including the windows, menus, popups, programs, etc., that are built and distributed as a single unit.

You might well have systems that require a much wider concept of an application. For example, we might consider each component of an accounting suite (receivables, payables, payroll, etc.) to be a separate application, because each can be built and distributed as an individual piece. But you might think of them as all belonging to one application, since the data in these modules is interrelated.

This difference illustrates why you must be quite clear about how applications in Advanced Revelation are built and maintained. Consider the example of the accounting suite above, and the advantages and disadvantages of separating each module into an Advanced Revelation application of its own. The advantages are those listed just above: security, ease of installation and maintenance, and ease of use. However, there would be some drawbacks as well:

- While life is much easier when you separate truly distinct applications, it requires extra effort if you need to share data between related tables in separate Advanced Revelation applications (although it's possible to share data between applications).
- Switching between one Advanced Revelation application and another can be impractical if you have to do it constantly in your everyday work with a suite of modules.

One way to get the best of both worlds is to create a single Advanced Revelation application for the entire suite, but to segregate modules by location. Advanced Revelation allows you to keep tables for an application on as many different locations as you want—and you don't have to make all locations accessible at the same time.

For example, you could create an application called ACCOUNTING. Within this application you would create all the tables for all the modules—receivables, payables, etc. However, the tables for each module would be on separate locations. The receivables tables might be on a subdirectory called AR, while the payables tables might be on a subdirectory called AP. Each module would therefore be a stand-alone unit, and could be installed and maintained by itself. However, as part of your application you could attach and detach the tables on these different locations as necessary.

The configuration of your applications is ultimately up to you, of course. But in laying out how and where you want the data to reside, you should consider the wide-scale implications of access, installation, maintenance, and ease of use, and develop accordingly.

Running an application

To run your application, users need their own copy of Advanced Revelation. There are several options here:

- If you want to restrict what the users can change in your application, you can provide them with a Runtime version of Advanced Revelation. A Runtime version lets them run an application but not make any changes to windows, programs, etc.
- If you intend to install your system at many different sites, you can also use a Runtime Deployment Kit, which allows you to produce your own Runtime versions of Advanced Revelation that you can distribute yourself.
- Users can also run your application using their own full development copy of Advanced Revelation. You might choose this option if you want to allow users to make their own changes or enhancements to your application.
- If you are working on the same network as your users, you can purchase a LAN Pack to turn your copy of Advanced Revelation into a multi-user version, and then allow users to share your copy. A LAN Pack increments the number of users for your copy of Advanced Revelation in groups of five.

Note You are not allowed (by the license agreement) to simply copy your version of Advanced Revelation to another machine. Each stand-alone computer or each network file server must have a copy of Advanced Revelation with a unique serial number.

The application development process

Each application is different, of course, but the basic steps to developing an application in Advanced Revelation are these:

1. Start a new application in Advanced Revelation. This builds a set of empty tables ("application system tables") in a location that you specify. These tables will eventually hold your windows, popups, menus, etc.
2. If you are concerned about security, you can assign a password to the application.
3. Define users for your application. Each user can have a different profile, including startup commands, passwords, security levels, and environment settings.
4. Open the application. As the developer, you will see the Development menu, from which you can access all the development tools.
5. Define data tables and the column definitions for them. As you will read later, we will recommend that all the data tables for an application reside in a single location that is separate from the tables for other applications and from the application system tables.
6. Create the windows, popups, menus, messages, etc., that the user will see as part of the application. All of these processes create entries in the application system tables that were created when you first began the application. For example, each time you create a data entry window, Advanced Revelation stores its definition in the table called WINDOWS, and each popup you define creates an entry in the table called POPUPS.
7. If your application involves programs (most do), create the program source code, then compile it. As with window and other tools, each program you create is an entry in a table.
8. Create any additional processes required for your application: indexes for the data tables, report definitions, etc.
9. Define the main and the subsidiary menus for the application. Instead of seeing the Development menu (as you do), the users will see this application main menu when they open the application.
10. If appropriate, define startup commands that access tables on non-default locations and launch application-specific processes when the application is opened.

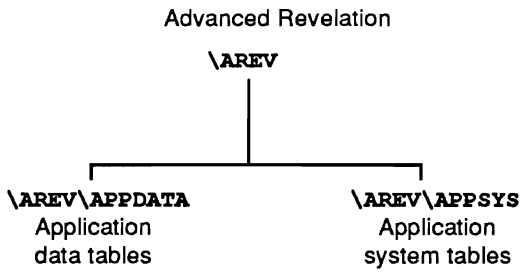
At this point you are done. If your users are on a network sharing your copy of Advanced Revelation, they can simply open the application and start using it. If the users have their own copies of Advanced Revelation, you can install the application onto their system (described later in this chapter).

Applications vs. locations

When you create a new application, Advanced Revelation asks you for a location. The system puts empty system tables in the location you provide. Advanced Revelation also stores the location as part of the application definition so that it knows to attach the tables on that location whenever you open that application.

You are by no means limited to just that location, however. An application can have tables on as many different locations as you want. Each time you define a new data table, you have the opportunity to specify a location for that table.

We recommend that you keep all the data tables for an application on one location. This makes it easier to find the tables, to manage them (such as backing them up), and to transfer them to another copy of Advanced Revelation:



We recommend that you keep the system and data tables for each application on separate subdirectories.

If you follow this strategy, you can also establish a default data location (see below) that simplifies maintenance.

You can use any path name when you are prompted for a location. However, think carefully about how the application will eventually be used. For example, you don't want to set up a location called D:\AREV\APPSYS if you anticipate that the application might be used later by people who don't have a D: drive. Whenever possible use relative path names—for example, rather than C:\AREV\APPSYS, you can use just APPSYS to indicate that the subdirectory is below your current Advanced Revelation subdirectory.

Accessing application tables

When you first open an application, the system looks in two locations for tables belonging to your application: the location you specified when you created the application, and the default data location. However, you may have elected to define tables for your application on other locations.

In that case, you will need to attach those tables before your application can use them. There are a couple of ways to do this:

- Manually attach the locations where the tables reside. Use the menu option **DB Admin-Tables-Attach** or the TCL ATTACHTABLE command.
- Set up a startup command that attaches the tables automatically when you open the application. See below for details.

If your application always attaches the same set of tables, you can greatly speed the process by using the SETATTACH command to create a "snapshot" of the tables you need. See the TCL SETATTACH command in the *Reference* book for more details.

Creating applications

Creating a new application

Before creating a new application, you will need to know:

- The name of the application you want to create. The name can be any length, and can include most characters except spaces and periods.
- The location for the application system tables. The subdirectory or path does not already have to exist.

To create a new application, follow these steps:

1. From the Advanced Revelation Opening menu, choose **Application-New**. The Application window displays.
2. At the *Name* prompt, enter a name for the new application.
3. At the *Location* prompt, enter the location where you want the application system tables to reside. By default this will be a subdirectory below your current location. The name will be based on the first eight characters of your application name.

   Change the name if you want to indicate a different location.
4. Press [F9] or click <Save> to start the application creation process.

When you save the definition, Advanced Revelation creates the new application by creating a number of tables on the location you specify. (You will see status reports in

the status line.) When the process is done, Advanced Revelation opens the new application for you and places you at the Development menu for the new application.

With the new application created, you might want to choose some options:

- Establish the default location of your data tables.
- Create new users for the application.
- Establish security for the application.
- Establish an application main menu to replace the Development menu.
- Create an alternate application shell.

Each of these options is described later in this chapter.

Accessing an application

To access an existing application, choose **Application-Open** from the Opening menu, then enter the name of the application at the *Name* prompt. Press [F2] or click <Options> to see a list of applications.

Alternatively, you can access the application directly when you log into Advanced Revelation. Add the application (or user) name after the word AREV when you log on from the operating system:

```
AREV MYAPP [/options]
```

If you open the application directly from the operating system , you bypass the Opening menu altogether and see only the Development menu (or the application main menu, if you've created one).

Quitting an application

When you are finished, you can quit the application in two ways:

- Close the application and return to the Opening menu.
- Quit and return to the operating system.

Choose the **Exit** option from the Development menu and select either **Close** or **Quit**. If you choose the option **Close**, Advanced Revelation will return to the SYSPROG application. If there is a password on the SYSPROG application, you will have to enter it before you can return to the Opening menu.

Related topic

- TCL OFF command

Application options

Setting up a default data location

When you create the application, you assign it a default location for the system tables in the application. We recommend that you establish a separate location for the data tables.

You always have the option of specifying a location when you create a data table. To make it easier to keep all data tables in one location, however, you may want to establish a default data location. To do so:

1. From the Advanced Revelation Opening menu, choose the option **Options-Configuration-Environment-General.**
2. At the window, press [Shift-F1] to see a list of applications. Choose the application you want.
3. At the *Default data location* prompt, enter the name of the drive or subdirectory that will be the default location for your data tables.
4. Save your changes with [F9] or by clicking <Save>.

Installing an application main menu

If you have created a custom main menu for your application, install it using these steps:

1. From the Opening menu, choose **Options-Configuration-Environment-Menus.** The Menu Environment window displays.
2. Press [Shift-F1] to display a list of applications. Choose your application from the list.
3. At the *Default System menu* prompt, enter the name of your application main menu.
4. Save your changes with [F9] or by clicking <Save>. The next time you open your application, Advanced Revelation will display the menu you've entered here.

Application security

Advanced Revelation provides application security in a number of ways:

- When you are in an application, users can't normally get access to tables in another if there is a password on the second application.

- Each application can have one or more passwords. If a user doesn't know at least one of the passwords, he or she will not be able to open the application.
- Each user you define for an application can have his or her own password. The user's password is checked if a user opens the application directly from the operating system.
- Each window and menu can have its own security level and set of passwords. When a user attempts to use a window or menu, his or her security level is checked against the corresponding security level.
- If you want to prevent users from changing windows, you can encrypt the windows so that users can't use Paint or other tools to change the windows.
- Finally, each window and menu can be assigned to one more applications so that only those applications can access the window or menu, even if other applications get access to it using an alias.

Caution If you assign a password to the SYSPROG application, keep a good record of it. If you forget the password, you will lose access to the SYSPROG application and will have to re-install your system.

Alternate passwords

There are two kinds of passwords for the application. The primary password is always checked first. However, you can also establish alternate passwords. The benefit is that you can link these alternate passwords to different startup commands and security levels. If a user opens the application with an alternate password, they might have a different security level, see a different main menu, etc.

Related topics

- TCL PASSWORD command

Security levels

You can assign security levels to both applications and users. A security level is a number from 1 (most permission) to 9 (least permission). No security level means that there are no restrictions.

Security levels are used in conjunction with windows (and menus). When you create a window, you have the option of assigning a security level and one or more passwords to the window. If the window has a security level, and if the current application or user has a security level, the two are compared. The user must have a security level at least as permissive as the window in order to access the window, unless the user knows one of the window's passwords. For example, if the window security level is 2, but the

user's security level is 3, the user can only use that window by providing one of the window's passwords.

You can set security levels for menus as well. Menu security works in the same way as window security as descibed just above.

You can assign a security level to an application as a whole. However, you are more likely to assign security levels to individual users. User security levels always take precedence over application security levels.

Adding or changing application passwords

To add or change the primary application password:

1. From the Opening menu, choose the option **Options-Applications**. The application window displays.
2. At the *Name* prompt, enter the name of the application for which you want to change the password. Press [F2] or click <Options> to see a list of applications.
3. At the *Password* prompt, enter a new password. As soon as you enter it, it will be replaced with the word "<encrypted>" so no one can see it. Save your change with [F9] or by clicking on <Save>.

You can also change a password from within the application. From the main menu, choose the option **Utilities-Password**, then enter a new password.

To remove a password

Follow the steps above, but don't enter a password at the *Password* prompt. Instead, go to that prompt and press [Ctrl-X] to clear out the prompt. The word "<encrypted>" will disappear to indicate that there is no password.

Adding or changing an alternate password

1. From the Opening menu, choose the option **Options-Applications**. The application window displays.
2. At the *Name* prompt, enter the name of the application.
3. Press [F10] to see the Application window menu. Choose the option **Options**. The Open Application Options window appears.
4. At the *Password* prompt, enter each of the alternate passwords. For each password, you can also enter:
   - The name of the startup command that you want to associate with that password.

- The security restriction level for that password.
- The Shell command for that password.

If you leave any of these prompts blank, the system will use the values established for the application as a whole.

5. Press [F9] to save your alternate passwords, and [F9] again to save the application information, or click on <Save>.

Setting an application security level

To assign a security level to an application as a whole:

1. From the Opening menu, choose the option **Options-Application**. The Application window displays.
2. At the *Application* prompt, enter the name of the application for which you are establishing a security level.
3. Press [F10] to access the Application window menu.
4. Choose the option **Advanced**.
5. At the *Restriction level* prompt, enter a number from 1 to 9. Save your changes with [F9] or <Save>, and then [F9] or <Save> again at the Application window.

Encrypting windows

A way of protecting windows from change (but not from access) is to encrypt them. An encrypted window functions just like an ordinary window, but cannot be changed.

The encryption process works by reading an existing window template (definition), encrypting it, and then saving the encrypted version. You can choose whether you want to store the encrypted and unencrypted versions of a window in the same table. You also determine what name you want to assign to the encrypted version of a window.

Caution Always be sure to keep an *unencrypted* version of a window available, because you cannot decrypt the window if you want to make further changes to it.

Generally, we recommend that you store the encrypted versions of windows in their own table. That way there is no possibility that you will overwrite the original, unencrypted versions of the windows.

To encrypt windows:

1. From the Development menu, choose **Developer-Encrypt**. *The Encrypt window* appears.

2. At the *Source table* prompt, enter the name of the table that contains the window definitions that you want to encrypt. Normally this would be the WINDOWS table.
3. At the *Destination table* prompt, enter the name of the table where you want to keep the encrypted versions of the windows. This can be the same name as the source table (but you will need to be very careful when assigning names to the encrypted windows!)
4. At the *Source row(s)* prompt, enter the names of the windows that you want to encrypt. Press [F2] or click <Options> to see a list of window definitions.
5. At the *Destination row(s)* prompt, enter the names that you want to assign to the encrypted windows, using a one-for-one correspondence with the names in the *Source row(s)* prompt.

Caution! Be very careful when assigning names if you are using the same source and destination *table*. If you assign the same name to the source and destination row, the encrypted version of the window will overwrite the non-encrypted version, and you will lose the unencrypted version!

6. At the *Options* prompt, enter one of these three options:

N	(New row inhibit.) Only save the encrypted version if there is already a template with that name.
O	(Overwrite.) Overwrite any existing template of that name.
S	(Suppress messages.) Post no messages during the encryption process.

Managing applications

Installing an application on another system

It's quite common to develop an application on one system and then copy or move it to a second copy of Advanced Revelation. The most obvious reason is to install your application so that others can use it on their copy of Advanced Revelation.

If you have followed our recommendations and segregated your application tables into one or two subdirectories, then the process is fairly simple. If you have intermingled tables from several applications in one location, you will need to isolate them before attempting to copy them to another system.

Note The process described below is *only* for version 3.0 and higher of Advanced Revelation.

In addition to copying the data and system tables, you will need to copy the application and user definition information. For example, if you have established several users and assigned special characteristics to each, you will want to copy this information to the new system as well.

The simplest way to do this is to create a temporary table and copy the application and user information to it. You must do this from the Advanced Revelation Opening menu, because you can't access application and user information when you are in an application. When you install the application on the second system, you can then copy this information from the temporary table to the correct location in the new system.

Assuming that you have segregated your application tables, the basic steps are these:

- Prepare your application to look and feel the way you want users to see it. For example, set up the application main menu to have on it the options the users will need.
- If you want to restrict changes to the application, you might want to consider these steps:

  1. Encrypting the windows so they can't be changed.
  2. Removing program source code from the system and leaving only object code.

  Be sure to make a complete backup of your application before doing either of these!
- Copy the user and application configuration information to a temporary table so that you can move it to the new system. The application and user information is located in the SYSENV table (a system table), so that it isn't stored with the rest of your application system information. We will suggest that you create a temporary table called APPENV that resides on the same location as your other application data.

To make a copy of your user and application information, follow these steps:

1. Log onto Advanced Revelation with no user or application name. You will see the Opening menu.
2. Press [F5] to display the Command window.
3. Create a the temporary table called APPENV by entering:

   ```
   MAKETABLE location APPENV
   ```

   where "location" is the location of your application.
4. Enter the word COPYROW. The Copyrow window displays.
5. At the *Source table* prompt enter SYSENV.
6. At the *Destination table* prompt enter APPENV (the name of your temporary table).

7. At the *Source row(s)* prompt, enter the names of these rows:
   - The name of your application. For example if your application is called PAYROLL, enter the name PAYROLL.
   - The name of your application plus the suffix "_ENVIRONMENT" (example: PAYROLL_ENVIRONMENT)
   - The names of any users that you want to copy to the new system. (Press [F2] or click <Options> to see a list of all the rows in the SYSENV table.)
   - The names of the environments for your users. The names will be the user name plus the suffix "_ENVIRONMENT" (example: FRANK_ENVIRONMENT). Be sure to include the correct environment entries if you have assigned environments to your users or to your application.
8. Press [F9] or click on <Save> to carry out the copy operation.

You can now copy the all the subdirectories that contain tables for the application to the new system. You can use any operating system command to do this, including COPY, XCOPY, or any backup utility.

Caution If you copy onto a subdirectory that already contains Advanced Revelation tables, *you will lose the data already there, and cannot recover it.* We recommend that you log first to the target directory and then copy *from* the source.

When you have finished copying the subdirectories to the new system, you can copy the application and user information to the new system by following steps very similar to those you used to copy it to the temporary table:

1. Log onto the new system without an application or user name. The Opening menu displays.
2. Press [F5] to display the Command window.
3. Make the temporary table available to the new system. Enter:

   ```
   ATTACHTABLE location APPENV
   ```

   where "location" is the location where you've put the temporary table containing the application and user information.
4. Enter COPYROW. The Copyrow window displays.
5. At the *Source table* prompt enter APPENV.
6. At the *Destination table* prompt enter SYSENV.
7. At the *Source row(s)* prompt, enter a star (*) to indicate that you want to copy all the rows from the temporary table to the SYSENV table.
8. Press [F9] or click on <Save> to start the copy process.

Note You may discover that the application or user entries already exist at the new site. If the application entry is already there, then that application (or one of that name) has already been created, and you should not proceed until you have resolved this conflict in names. If you find that users already exist, you can choose to overwrite them by entering "O" at the Options prompt in the Copyrow window. Be sure, however, that you are not overwriting important information in the new system.

9. If you like, you can remove the APPENV table from your system, since you won't need it any more. At the Command window enter:

```
DELETETABLE APPENV
```

When the copy process is done, you have finished copying the application. You can then access the application on the new system by opening the application (**Application-Open** from the Opening menu in Advanced Revelation) or by typing AREV *application* at the operating system prompt to log onto Advanced Revelation.

If you have intermingled tables from different applications in one location, you will need to isolate them. The best strategy is to copy the system tables for your application to one location, and the data tables to another. Determine first where you want the tables to reside. If necessary, create a new subdirectory at the operating system. Then follow the steps for copying a table (see the chapter "Managing tables"), in each case copying the table to the location you've just chosen. When you are done copying all the tables to a central location, follow the steps above for copying them to another copy of Advanced Revelation.

Note For information about system tables, see the Appendix "System tables" in the *Reference* manual.

Backing up an application

You can easily back up your application using operating system commands such as the operating system command BACKUP or using a commercially-available backup utility. It's possible to back up your entire Advanced Revelation system each time you do a backup, but this often isn't necessary. For example, the basic Advanced Revelation system (the information on the subdirectory where you installed Advanced Revelation) rarely changes. Instead, you should concentrate on backing up individual applications.

To back up an application, follow the steps for installing an application on another system (see above) to the point where you have copied the subdirectories to another location (step 8). Then copy the subdirectories to a safe location.

The steps above instruct you to copy the application and user information to a temporary table. You should do this as part of the backup also. However, if you have not changed this information, you do not need to keep repeating this step—but do be sure to save this information at least once.

Backing up individual tables

If you want to back up an individual table, you can use Advanced Revelation's table management tools to copy the table to a new subdirectory, and then use an operating system utility to back up the subdirectory.

Restoring an individual table

If you back up this way, you need to be careful if you ever need to restore the data. Don't restore directly onto the original location! Instead, follow these steps:

1. Restore the subdirectory onto a temporary directory somewhere on your system.
2. Log onto Advanced Revelation and open the application for which you are restoring the table.
3. Attach the temporary subdirectory using the option **DB Admin-Table-Attach.**
4. Using the option **DB Admin-Table-Copy**, copy the restored table back to the original location. You must enter "O" (overwrite) at the *Option(s)* prompt.
5. Using the option **DB Admin-Table-Attach**, attach the original location.

Removing an application

If you are finished with an application, you can remove it entirely from your system. Advanced Revelation removes all the tables for an application as well as the application definition information. This process does not remove users from the system, even if they are associated with the application that you are removing. (You can assign these users to another application, or remove them separately.)

By default, Advanced Revelation will remove all the tables on the location associated with your application (tables from other applications that happen to reside in that location will be left alone). This does *not* include the default data location. You have the option of telling Advanced Revelation in what additional locations you are maintaining tables that belong to the application.

Caution Once you remove an application, you cannot recover the data in it, so be sure to take precautionary steps before deleting.

To remove an application:

1. Close the current application or log onto Advanced Revelation with no user or application name. The Opening menu displays.
2. Choose the option **Options-Application.** The Application window displays.
3. At the *Application* prompt enter the name of the application you want to remove.
4. Press [F10] to see the Application window menu.

5. Choose the option **Delete**. The Delete window displays.
6. At the *Additional locations* prompt, enter the names of any locations other than the application location where you have application tables.
7. Press [F9] or click on <Save> to to remove the application.

About updating applications

After you have installed an application on a user's computer, you might do further development. If so, you will have to transfer the changes from your system to the user's system. Depending on the nature and thoroughness of your changes, and the likelihood that the user has made custom changes to his or her system, you have a couple of options:

- You can completely overwrite the user's application *system* subdirectory, replacing all the tables with the ones from your system. This is a good option if the user's application system tables are exactly the same as yours, and he or she is not allowed to change or customize them.

  Caution! *Never* overwrite any location (subdirectory) containing the user's data! If the user maintains data tables on the same location as the application system tables, do *not* copy your subdirectory as a unit. If you do, the users will lose all the data (and other changes) on that subdirectory permanently!

- You can copy individual entries from your system to theirs. This would be the best strategy if you want to update only one window, one program, or one dictionary entry. This is also the safest strategy if you are not sure whether you would be overwriting important user tables by copying an entire subdirectory. However, it requires more work on your part, and puts a burden on you to copy individually every changed component in your system.

Updating an entire subdirectory

If you want to replace *completely* a user's application system tables, simply copy your own version of the application system location from your computer to the user's computer using an operating system command or backup utility. If you want to restrict access to parts of your system (for example, to windows or R/BASIC code), encrypt the windows or remove the source code before copying the directory (make a backup of your own system before encrypting windows or removing program source code).

Caution! Be absolutely sure that the user does not have data tables on the target directory, and that the user has not made custom changes to their system. Both of these will be lost without the possibility of recovery if you overwrite them by copying an entire directory.

Updating individual components in an application

If you want to update only individual elements of the application—windows, programs, menus, column definitions, R/BASIC programs, etc.—you can copy these one-by-one to the user's system. Before you do this, you will need to know:

- What tables in your system contain the elements you want to update. For example, window definitions are typically stored in a table called WINDOWS, xmenus are stored in a table called MENUS, etc. Sometimes a single change in your system results in changes to several tables. For example, if you create a new R/BASIC subroutine, you will need both the object code *and* the catalog entry from the VOC table.

  For information about system tables, see Appendix 1, "System Tables".

- Which elements you have changed. It will be up to you to know what changes you have made to your system—what windows have changed, what menus, what column definitions you've updated, what R/BASIC programs you have created or changed. Remember, for example, that if you index a column, the column definition has been changed, and will need to be copied to the user's system.

  There is currently no utility in Advanced Revelation that can track this information for you.

To copy individual changes to a user's system:

1. Create a new, temporary subdirectory on your system.
2. For each table you've updated in your system, create a corresponding temporary table on the new directory. For example, if you have modified a window, you have updated the WINDOWS table. You will therefore need a table called TEMP_WINDOWS on the new directory. If you want to copy changed column definitions, create temporary tables for them with a name such as TEMP_DICT_*tablename*.
3. Using the row management tools, copy the individual changes to the temporary tables. For example, you can copy an updated window definition from the WINDOWS table to the TEMP_WINDOWS table, a new menu from MENUS to TEMP_MENUS, etc. If you are updating R/BASIC programs, remember to copy the object code version (the entry with the '$' prefix) to the temporary table, and if necessary, the catalog entry from the VOC table.
4. When you have copied all changed elements from your system to the temporary directory, transfer the temporary directory to the user's computer by copying it to a new temporary directory there.
5. Log onto the user's system and open the application you want to update.
6. Attach the temporary directory that you just copied to the user's computer.

7. Copy the entries from the temporary tables to the corresponding system tables in the application. For example, if you want to update a window definition, copy the definition from the TEMP_WINDOWS table to the user's WINDOWS table. Use the "O" (overwrite) option when copying the entries.

About users

You can define many users for a single application. Each user can have his or her own password, startup command, and environment settings. By defining users, you can make your application look and behave differently for each person using the application.

Having multiple users for an application gives you great flexibility in how you allow users to access your application. For example, consider these variations:

- User Frank, a "power user", sees a full application menu that includes ad-hoc reporting options. He also has access to all application tables.
- User Mary cannot use certain windows because her security level doesn't give her permission to.
- User Andrew is like user Frank, but his startup command doesn't make available all possible application tables.
- User Guest is restricted from using [F5] to access the Command window.

For more information about startup commands, see that topic later in this chapter.

The SYSPROG user

If you close an application or log onto Advanced Revelation with no application or user name, you will be in the special application called SYSPROG. You will have the special user name SYSPROG. The SYSPROG user has no security restrictions, so you might want to be careful about allowing users to access this user.

The default user

If you begin at the Advanced Revelation Opening menu and open an application, you will not have an opportunity to tell the system what user name you want to use. Instead, Advanced Revelation will open the application and assign you a user name that is the same as the application name. For example, if you open the application PAYROLL from the Opening menu, you will have the user name PAYROLL.

If you are using the default user in an application, the characteristics (environment) for that user are the same as those that you define for the application as a whole.

Creating a new user

Before you create a new user, you will need to know:

- The name of the user. User names must be unique throughout your copy of Advanced Revelation (not just for a single application), and can contain no spaces or punctuation.
- What application the user will be assigned to.
- Whether you want to have a password for the user.

To create a new user:

1. From the Opening menu, choose the option **Options-Users**. The User window displays.
2. At the *User name* prompt, enter the name of the new user.
3. At the *Application* prompt, enter the name of the application that this user is assigned to.
4. At the *Password* prompt, enter a password for this user, if any. After you've entered it, the word "<encrypted>" will appear to let you know that there is a password for this user.

Accessing an application using a user name

To access an application using the user name, you must add the user name to the word AREV when you log onto Advanced Revelation from the operating system prompt. Advanced Revelation will access the application that the user is assigned to. For example, to log on using the user name FRANK, you would enter this at the operating system prompt:

```
AREV FRANK
```

Adding or changing a user password

1. From the Opening menu, choose the option **Options-Users**. The User window displays.
2. At the *Name* prompt, enter the name of the user for which you want to change the password.
3. At the Password prompt, enter the new or changed password. As soon as you do the word "<encrypted>" appears so no one can see the password.

To remove a password

Follow the steps above, but don't enter a password at the *Password* prompt. Instead, go to that prompt and press [Ctrl-X] to clear out the prompt. The word "<encrypted>" will disappear to indicate that there is no password.

Setting a user security level for a user

1. From the Opening menu, choose the option **Options-Users**. The User window displays.
2. At the *Name* prompt, enter the name of the user for which you are establishing a security level.
3. Press [F10] to access the User window menu.
4. Choose the option **Advanced**.
5. At the *Restriction level* prompt, enter a number from 1 to 9. Save your changes with [F9], and then [F9] again at the Application window.

Assigning a user to a different application

When you first create a user, you usually assign the user to an application. However, you can change the user's application at any time. If you do so, all the user's characteristics (password, security level, etc.) apply in the new application. Be sure that the user's environment (example: startup menu) is correct for the new application as well. To change the application for a user:

1. From the Opening menu, choose the option **Options-Users**.
2. At the *Name* prompt enter the name of the user.
3. At the *Application* prompt, enter the name of the new application. Save your change with [F9].

Removing a user from the system

1. From the Opening menu, choose the option **Options-Users**.
2. At the *Name* prompt enter the name of the user.
3. Press [F10] to display the User window menu, then choose the option **Delete**.

Environments for applications and users

An environment is a collection of attributes that you can assign to a user or to the application as a whole. Characteristics that can be set in an environment include:

- The starting menu.
- The default location for data tables.
- Whether the user sees a status line or not.
- Access to the Command window.
- The ability to create or use macro keys.
- Whether the user sees a menu or the Command window when opening an application.

When you create a new application or user, Advanced Revelation automatically creates an environment to match. This environment has default values that are the same for all new applications or users.

When you open the application, Advanced Revelation looks for the corresponding environment and uses the values it finds there. If you make changes to the environment, those changes will affect everyone who opens the application from the Opening menu.

The same is true for users. If you log onto Advanced Revelation using a user name, the system finds the corresponding environment and uses those values. As with the application environment, if you change the user's environment, it affects everyone who logs onto Advanced Revelation using that user name.

Although environments are created at the same time as the application or user, they are not tightly bound together. You can assign an environment created for one user to another user so that both users have the same characteristics.

Changing an environment

The basic steps for changing any environment setting are always the same, whether for an application or for a user:

1. Close the current application or log onto Advanced Revelation with no user or application name.
2. From the Opening menu, choose **Options-Configuration-Environment**. The Environment menu displays.
3. Choose the option that describes the change you want to make (example: General). The corresponding environment window displays.

4. Press [Shift-F1] to see a list of all possible environments. Choose the one you want to change.
5. Make your change in the window and save it with [F9] or by clicking <Save>.

As you can see from the choices in the Environment menu, there are many things you can customize for each environment. A few of the most common have already been listed earlier (changing the default data location, assigning a menu to an application or user). A few more are listed below. If you want more detail on the environment settings you can change, press [F1] at any prompt in any environment window.

Restricting access to the Command window

1. From the Opening menu, choose **Options-Configuration-Environment-General.**
2. Press [Shift-F1] to see a list of environments. Choose the environment you want to change.
3. At the *Enable TCL Key* prompt, enter "N". Save your changes with [F9].

Turning off the status line

1. From the Opening menu, choose **Options-Configuration-Environment-General.**
2. Press [Shift-F1] to see a list of environments. Choose the environment you want to change.
3. At the *Display status line* prompt, enter "N" to turn the status line off. Enter "Y" to turn the status line back on. Save your changes with [F9].

Starting up in menu vs. command mode

1. From the Opening menu, choose **Options-Configuration-Environment-General.**
2. Press [Shift-F1] to see a list of environments. Choose the environment you want to change.
3. At the *Default system mode* prompt, enter "T" to start up in command mode (TCL) or "M" to start up in menu mode. Save your changes with [F9].

Assigning an environment to an application or user

If you want to have an application or user use an environment other than the default environment, you can assign a different one. You would do this, for example, if you wanted several users to share the same environment definition. To do this:

1. From the Opening menu, choose the option **Options-Applications** or **Options-Users**.
2. At the Application or User window, enter the name of the application or user to which you want to assign a new environment.
3. Press [F10] to display the window menu, and choose **Advanced**.
4. At the Environment prompt, enter the name of the environment you want to assign to that application or user.
5. Press [F9] or click on <Save> to save changes, and then again at the Application or User window.

Startup commands

Each application and even each user and alternate password can have its own startup command. The startup command is run after you open the application and before the application main menu displays.

A startup command can be any command, or any series of commands, that you could execute at the Command window. Common uses for startup commands are to attach tables in different locations or to run custom security checks.

Startup commands are actually scripts that are stored in the VOC (vocabulary) table. Creating a startup command is therefore a matter of defining an entry in that table. To learn more about TCL, see the *Reference* book. Note especially the Batch commands.

When you open an application, Advanced Revelation always looks for four possible startup commands, in this order:

- The startup command that you explicitly associate with an application or user.
- A startup command with the same name as the current user, if a user name was used to open the application.
- A startup command with the same name as the current application.
- Finally, a startup command named LOGON.

To create a startup command:

1. Open the application for which you are defining the startup command.
2. In the application, press [F5] to display the Command window.
3. Press [Shift-F1] to display the Link window.
4. At the *Command(s)* prompt, enter the commands you want to run as part of your startup command, one command per line.

5. At the *Save key* prompt, enter the name of the startup command. This can be the name of the application or of the user, the word LOGON, or a custom name of your own.
6. Save the startup command with [F9].

Installing a startup command

If you assigned the application or user name or the name LOGON to the startup command, you are done, because the system will find the startup command automatically when you open the application. However, if you have assigned a custom name to the startup command, install it on the application using these steps:

1. Close or leave the application, if necessary, and return to the Advanced Revelation Opening menu.
2. Choose **Options-Applications**. The Application window displays.
3. At the *Name* prompt enter the name of the application for which you are installing the startup command. The application information displays.
4. Press [F10] to display the Application window menu, then choose **Advanced**.
5. At the *Startup command* prompt, enter the name of the startup command that you created above. Be sure the name you enter here matches the name that you assigned to the startup command.
6. Press [F9] or click on <Save> to save your option, then again to save the application information.

To install a startup command for a user, follow the steps for installing a startup command for an application, but instead of choosing **Applications** from the Opening menu, choose **Users**.

To install a startup command for an alternate password, follow the steps for installing it on an application. However, instead of choosing **Advanced** from the Application window menu, choose **Options**. At the *Password* prompt move to the password for which you want to install the startup command (or enter a new password), and at the Startup command prompt enter the name that you assigned to the startup command. Save changes in both windows with [F9].

Application shells

When you open an application, it will first run the startup command (if there is one), and then display a main menu. However, you have the option of creating your own application shell. In that case, the system will run the startup command, but then call your process as the "shell" for the application.

Shells are often used if the application has a completely custom interface and doesn't use the default Advanced Revelation menus. For example, you might have your own menu or security system that you want in place of the current one. As another example, you might want to jump directly into a program that runs a batch process, then logs off.

You can install a shell for an application as a whole. In that case, all users of the application use the same shell. Alternatively, you can assign a shell to an individual user or to an alternate password. The latter two choices allow you to use the shell under special circumstances, such as the example above where the shell executes a batch process.

Although you can designate any Advanced Revelation tool to be a shell, a shell is very frequently an R/BASIC program. This allows you to take complete control over your application's operations.

When you install a shell, it becomes the top-level process in the application, and remains permanently resident while the application runs. All processing will begin and end in your shell, so you should create it with this in mind, and make it an efficient process that uses memory sparingly.

To install an application shell:

1. From the Opening menu, choose **Options-Application**. The Application window displays.
2. At the Name prompt, enter the name of the application for which you are installing the shell. The application information displays.
3. Press [F10] to see the Application window menu.
4. Choose **Advanced**. The Advanced window displays.
5. At the *Shell code and command* prompt, enter a code and command that calls your shell. Almost always this will be a code of "S" (for subroutine) and a command that names the R/BASIC routine you wrote to serve as the shell.
6. Save your options with [F9] or by clicking <Save>, and then again to save the application information.

To associate an application shell with an alternate password:

1. Follow steps 1 through 4, but instead of choosing the option **Advanced**, choose the option **Options**.
2. At the *Password* prompt, enter an alternate password, or select the alternate password you want to use. Move to the *Shell code and command* next to the password you want to use, and enter the code and command for the shell.
3. Save your changes in both windows by pressing [F9] or clicking <Save> twice.

8. Creating windows using Paint

About Paint

Advanced Revelation's Paint is a menu-driven tool for creating and modifying application windows quickly without writing source code. You don't need extensive programming knowledge or expertise with Advanced Revelation to use Paint. Paint makes it easier for you to:

- Create and modify **data entry windows**
- Create and modify **collector windows**
- Create and modify **prompts** in windows
- Define the characteristics of prompts (called **prompt details**)
- Create and modify screen text (called **labels**)
- Set up validations and formats for input data
- Run tests of the windows you create

Without leaving the Paint menu environment, you can also do such things as:

- Link windows to popups for look-ups and valid data selection
- Modify data tables and data dictionary definitions
- Link windows to menus
- Control colors of windows and window objects

Paint removes the chore of programming windows and related processes from the shoulders of the developer and uses the resources of the computer instead. Experienced developers can develop more and better applications. New developers can do productive work quickly—even before acquiring an expert level of knowledge of Advanced Revelation.

Developers often just need to create a user-friendly "front-end" that allows end users to add, delete, edit, and locate rows in a database easily and productively with certain basic safeguards to reduce data entry errors. Using Paint for this work is faster and easier than programming.

Before beginning with Paint

While you don't have to be an Advanced Revelation "guru" in order to use Paint, you do need to have some basic knowledge in several areas:

- Be familiar with common relational database terms like tables, rows, and columns.
- Be familiar with the tables in your application, including their column definitions and the relations between them.

- Have a basic plan for how your windows will look and operate.
- Anticipate any restrictions in formatting and validation that you will want to place on data entry prompts.
- Understand Advanced Revelation concepts such as codes & commands, and formulas.
- Be familiar with basic Advanced Revelation terminology. Understand such terms as applications, users, indexes, and locations. If you will be working with data from an environment other than Advanced Revelation, you need some knowledge of Environmental Bonding.

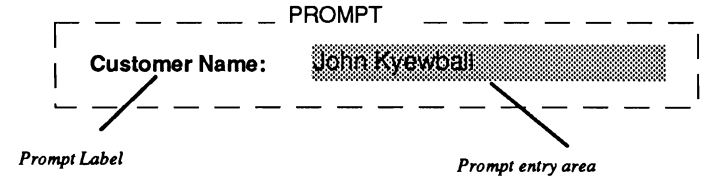
Paint terms and documentation conventions

Window: A data entry or display screen. There are two basic types: Data entry and Collector.

Data entry window: Displays data from or writes data to a data table. You will probably create this type of window most frequently.

Collector window: Collects information that will be used in an R/BASIC process.

Prompt: Sometimes called a field label or data label in other database systems. In Paint, a prompt consists of the prompt label followed by the entry or display area:



Label: Text that you place in a window to provide information to the user. (Example: "Press F1 for Help")

Binding: The process of connecting a prompt in a window to a column in a table so that the window can read data from it, or write data to it.

Prompt Details: Characteristics and restrictions associated with a prompt. Examples of prompt details are data type, justification, edit mask, output format, help text, etc.

Typographical conventions

Highlighted letters on menu options appear in **bold** type throughout the Paint section of the documentation.

Example: Edit-Paste
When the action bar menu is active, typing **e** chooses the Edit option. Typing **p** then chooses the Paste option from the related pull down. [F10] toggles the active state on and off.

About icons

Throughout the Paint documentation, icons symbolize the two ways—keyboard and mouse—that you can interact with the computer. The icon arrangement differs depending upon whether they represent a single action or a series of steps.

Icon format for a single action

An instruction for a single action is followed by the keyboard and mouse instructions:

"Save your window to disk: ( = [F9]  =Window-Save)"

This means that you can either press [F9] on the keyboard, or use the mouse and click first on Window, then on Save, to perform the interaction.

Note You could also type **t s** on the keyboard if the option is from an active menu. In this case, the [F9] shortcut would be faster. We will suggest shortcut keys wherever possible.

Icon format for series of steps

Where appropriate, we show the steps using keyboard, followed by the same steps using a mouse:

"The first task is to open the window you want to modify."

1. Press [F10] to toggle the menu on.
2. Select Window-Open.
3. Enter the table name and the window name in the Open Window popup.
4. Press [F9].

1. Select Template-Open *(options from an action bar menu and pulldown)*
2. Enter the table name and the template name in the Open Template popup
3. Select <Save> *(remember that the < > symbols indicate the status line)*

About navigation in Paint

The section on navigation teaches you how to move the cursor and select menu options, and covers the standard Advanced Revelation keystroke commands. You can use either the keyboard, a mouse, or other pointing device, or both to work in Paint. You should be familiar enough with the navigation methods available on your system to follow instructions in the documentation.

Terms related to navigation

Select. Use the keyboard or mouse to highlight one or more objects in a window. You can move, resize, edit, delete, attach processes, definitions, etc., to objects when they are in the selected state.

Deselect. Use the keyboard or mouse to highlight one or more objects in a window. You can move, resize, edit, delete, attach processes, definitions, etc., to objects when they are in the selected state.

Drag. Use keyboard or mouse to move a selected object on the screen.

Draw. Use keyboard or mouse to select the Draw option and to draw a box or line on the screen.

Outline. Drag a selecting outline around objects in a window.

Keystrokes and their behavior

You should be familiar with the navigational keystrokes that are standard in Advanced Revelation (see the chapter "Navigating Advanced Revelation"). Some of these, like [F1] Help and [F2] Options, behave the same way in Paint. Other keystrokes are specific to Paint. The following tables divide the available keystroke options into several categories and describe their functions within the Paint environment.

Finding options and help

If you want to...	Press
Display context-sensitive on-line help	[F1]
Display a popup of options that are available in the current edit mode, or for currently selected objects.	[F2]
Display a popup of the currently available softkeys, [Shift-F1] through [Shift-F10]	[F6]
Display a popup showing which keys or key combinations are currently active, along with a brief description of their function	[Ctrl-F9]

Editing, saving and escaping

If you want to...	Press
Toggle the Editor on and off	[F4]
Save the current window to disk (may also function as a "go ahead" or "OK" command)	[F9]
Toggle between Make Prompt mode (typing in the window creates a new prompt) and Label Edit mode (typing in the window creates a label)	[Shift-F2]
Display a window allowing you to view or edit basic prompt detail definitions	[Shift-F5]
Display a window allowing you to view or edit complete prompt detail definitions	[Shift-F6]
Display a popup listing current window's prompts *in the order the cursor moves through them*	[Shift-F7]
View or edit the dictionary window	[Shift-F10]
Display the Relations window to create joins between tables	[Ctrl-F6]
Edit colors for selected objects	[Ctrl-C]
Exit the current window	[Esc]

Selecting, moving and resizing

If you want to...	Press
Drag (move) selected objects to another place in the window	[Shift-F3]
Enable window resizing from the keyboard	[Ctrl-F7]
Move the window	[Ctrl-F8]
Select a prompt with the keyboard which is not near the cursor	[Alt-A]
Toggle the select status of objects in the current window	[Space Bar]
Move the cursor forward and backward through the prompt order, landing on each object in turn	[Tab] [Shift-Tab]
Move the cursor left, right, up, and down in full page increments	[Home] [End] [PageUp] [PageDown]
Move the cursor to top right, bottom right, top left, and bottom left corners of the window, including any portions outside the immediate field of view	[Ctrl-PgUp] [Ctrl-PgDn] [Ctrl-Home] [Ctrl-End]

Drawing, inserting and deleting

If you want to...	Press
Copy selected objects to the global Paint clipboard	[Ctrl-F3]
Paste current clipboard contents into the window at the current cursor position.	[Ctrl-F4]
Delete the current window *and all associated help entries* from disk	[Alt-D]
Draw boxes or lines in the current window	[Ctrl-B]
Delete any selected objects from the window permanently	[Del]

Finding and replacing

If you want to...	Press
Pan your view through the current window's virtual space	[F7]
Quick Paint prompts, using existing dictionary, into the window *in the order you select them,* starting at the current cursor position	[Shift-F9]
Search the window for a string of characters	[Ctrl-F]
Repeat a search using the same values as last search	[Ctrl-A]

Other keys

If you want to...	Press
Refresh the current display screen	[F8]
Activate the Paint action bar menu at the top of the screen	[F10]
Test run the window, optionally saving test rows.	[Shift-F1
Activate prompt-to-table binding	[Shift-F4]

Creating windows

The two most common situations you face when creating a new data entry window are:

- The new window will update an existing table with a defined dictionary.
- You must create a new table to go with the new window.

We will focus Paint documentation on creating windows when there is an *existing* data table. As part of the Paint start-up process you have the opportunity to create a new table for a new window. If you do, you are working with an existing table from that

point forward. For more information on creating a new table, see the chapter "Managing tables".

About binding and binding modes

During the Paint's start-up routine, it asks you first if you are creating a new window or accessing an existing one. Then, if you are creating a new window, you have the opportunity to create a new table.

You proceed to name the dictionary, specify the volume, and select the desired table from a popup of available tables. Paint enters **Autobind** mode, and will display "Bind On" on the status line. For every prompt you create, Paint will guide you through selection of a table column and bind the prompt to it.

If you respond "No" when prompted about binding, you are then essentially creating a **collector window**. Prompts you create do not bind to a table, and the status line will display "Bind Off". Collector windows mainly collect and hold input data for R/BASIC programs. Consult the R/BASIC documentation for more information.

It is possible to bind prompts to a table later if necessary. For information, see the topic "Managing tables and columns".

Creating a new window and dictionary definitions

Follow these steps when you are starting from scratch to create a data entry window.

1. Select Define-Window from the Development menu.
   (⌨ = d w 🖱 = Define-Window)
2. Paint asks you if you are opening a new, existing, or collector window. Respond "New".
3. Enter the dictionary and volume names in the Make Dictionary window.
4. Select Save to proceed. (⌨ = [F9] 🖱 = < Save >)
5. Select the new table name from the Tables Attached popup.
6. Respond "OK": (⌨ = [Enter] 🖱 = < OK >)
   - Paint displays a blank work area and is ready to for you to begin creating prompts. Typing any character will begin creating the first prompt in the window. Paint is in Autobind mode.

   Note The first prompt you enter into a new window should be the *key* prompt. While it is possible to redefine the key prompt later, it is much easier to enter the key prompt first. If other prompts will be part of the row key, enter them next.

- If you want to activate the Paint menu, press [F10] or click the mouse anywhere on the menu.

7. Proceed to create prompts and any other objects in the window.
   - If you need more information about creating or editing prompts, labels, lines or boxes, see these topic "Managing window objects". If you need information about window operations (moving, sizing, testing, customizing, etc.), see "Managing windows".
8. Save the new window to disk. (= [F9] = < Save >)

Creating a new window with QuickPaint

In Paint, you can either create prompts one at a time, defining their dictionary definitions individually, or you can use Quick Paint to create a whole screen fast. The Quick Paint option lets you create a number of prompts quickly using an existing table's dictionary. You pop up a list of the columns in the table and select the ones you want to use in the window—any or all of them. Quick Paint creates a layout using the default dictionary definitions. You can then rearrange prompts in the window, draw boxes around objects, add text labels, and redefine the dictionary attributes.

Note You need to consider your screen layout carefully before selecting columns, especially if you plan to select all columns in the table. You also need to consider your data dictionary definitions because multivalued prompts display on more than one line.

If you select more data than will fit in the available lines in the window, you will get a jumbled display. In that case, create two windows or a window with multiple pages (see "Managing windows").

To create prompts using Quick Paint:

1. Follow the previous steps 1-6 to create a new window. Select an existing table with an existing dictionary.
2. Select the Quick Paint option from the Paint menu:
   (= [Shift-F9] = Layout-Quick Paint)
3. Select the columns that you want to become prompts in the window from the popup of available columns.
   - *Use the keyboard.* Press [Enter] to select or deselect, and arrow keys to move to next selection. Selected items change color or highlight.

     Note The columns will appear selected in the window in the order you select in them in the popup. You can always rearrange the screen layout, change the order of the prompts, edit prompt details, add prompts, or perform any other prompt operations later.

4. Confirm your selections: (= [F9] = < OK >)
   - Unless you selected more columns than the screen can contain, you will see an orderly window layout with all the prompts selected.

Creating prompts

Creating basic prompts

Paint defaults to a state where it is ready to create prompts. To begin a new prompt:

1. Enter the characters of the prompt label. (Make-prompt mode displays on the status line.)
2. Press [Enter]. Paint creates the display-entry area of the prompt with a default length of 10 characters. Paint selects the prompt and activates the Editor (indicated on the status line).
3. Change the size of the data entry area with [←] and [→] keys, or press [Enter] to deactivate the Editor and return to the ready state.
4. Repeat 1, 2, and 3 for any additional prompts you want to create.

Note If Paint is in Autobind mode, it will guide you through the selection of a table column to which it will bind the prompt between steps 2 and 3.

Creating the key prompt

The first prompt you enter into a new data entry window is the **key prompt.** The key prompt corresponds to the table's row key column when the prompt binds to a table.

You create a key prompt in the same manner as a basic prompt. The only difference is that Paint assumes that the *first* prompt you enter is the key prompt and asks you to confirm this.

It is possible to redefine the key prompt later, but we strongly recommend that you know what your key prompt will be ahead of time and enter it first. If more than one prompt will be part of the key, enter other key prompts next. For further information on multipart keys, see "Creating multipart key prompts" elsewhere in this chapter.

Creating a multivalued prompt

A multivalued prompt corresponds to a multivalued column in the table to which it is bound (see the chapter "About databases in Advanced Revelation"). To create a multivalued prompt, you expand the size of the default data entry area after creating a basic prompt. To create a multivalued prompt:

1. Create a basic prompt as described previously.
2. Make sure the Editor has been activated with [F4]. Selected objects will appear in reverse video when the Editor is active.
3. Use [→] to expand, or [←] to contract the data entry area to the desired width.
4. Use [↓] to increase the number of lines in the data entry area. The [↑] key removes added lines. Each line will display one value of the multivalued column. As soon as there is more than one line for the prompt, Paint assumes that the prompt will be multvalued.

Note Keep in mind that data will scroll through a multivalued prompt's data entry area. You don't necessarily need to allow display space for the maximum possible number of lines. You can also limit the number of lines that a user can enter into a multivalued prompt.

Creating a text prompt

If you want to create a prompt that will accept paragraphs of text, follow these steps:

1. Create a basic prompt as described previously.
2. Use [→] to expand, or [←] to contract the data entry area to the desired width.
3. Use [↓] to increase the number of lines in the data entry area. The [↑] key removes added lines. A text prompt will have at least two lines.

   Note Keep in mind that data will scroll through a multivalued prompt's data entry area. You don't necessarily need to allow display space for the maximum possible number of lines. You can also limit the number of lines that a user can enter into a multivalued prompt.
4. If you are creating a prompt that will bond to a data column, bind to the column as you would normally.
5. With the prompt still selected, press [Shift-F5] or click <Details> to display the Prompt Detail window.
6. At the Justification prompt, enter "T" for a formatted text prompt (line breaks are maintained), or "TN" for a text non-justified prompt (formatting not maintained). Save your change with [F9] or <Save>.
7. If you are binding this prompt to a data column, answer "Y" when Paint asks whether you want to update the data column.
8. To update the data column, select 6-"Justification" from the Dictionary Attributes popup. Save your change in this popup by pressing [F9] or clicking <OK>.

Associating multivalued prompt groups

A multivalued prompt can be grouped together with other multivalued prompts in the window. The multivalued prompt group becomes permanently linked so that an action taken on one prompt—a delete, for example— affects others in the group. The process of creating this link is called **associating**, and the linked prompts become **associated multivalued prompts.** A group of these is called an associated multivalued group (**AMV Group**).

You can create one or more associated multivalued groups. You assign each separate group an **AMV Group number** at the time you create the AMV group.

To create an AMV Group:

1. Select Advanced-Groupings from the Paint menu:
   (= [F10] a g = Advanced-Groupings)
2. Type an AMV Group number in the grouping popup. This number becomes the current group for prompt selection and it will appear in the AMV Group column heading.
3. Select the prompts you want to be part of the group. The group number will appear in the AMV Group column for the selected prompt. [Enter] selects and deselects prompts.
4. Repeat steps 2 and 3 to create additional AMV Groups. Use a different number for each group. Change a prompt to different AMV Group by selecting it once to remove its group number, and again to change it to the current group number.
5. Save the groupings. (= [F9] = < OK >)

Creating multipart key prompts

Usually, there is only one column in a table, the key column, that uniquely identifies a row. For example, the CUST_CODE column in a Customer table. Sometimes you need more than one key to identify a row. An inventory row might need ITEM_CODE, VENDOR_CODE and DATE_RECVD to uniquely identify it.

You normally set up multipart keys when creating a new window. Paint helps you with this process by asking you if each new prompt is part of the key. Paint makes each new prompt part of the key until you indicate that the key is complete. To create a multipart key:

1. Open a new window, creating the new table if necessary.
2. Enter the prompt that will be the first part of the key.
3. Respond "Yes" when asked if the prompt is part of the key, and bind the prompt to a column as normal.

4. Enter prompts for the other parts of the key, responding "Yes" each time when Advanced Revelation asks if this is part of the key.
5. When you have created all the key prompts, begin entering non-key prompts.
6. Respond "No" to the key-part inquiry when entering the first non-key prompt. From then on, Paint will assume that you have defined all parts of the key.

Managing window objects

The following basic technique for managing prompts and labels are essential to most other functions that you can accomplish in Paint.

Selecting objects

Windows are composed of different objects—prompts, labels, lines, and boxes. In order to move or change an object, you must first **select** it. Selected objects appear highlighted on the screen.

Selecting a single prompt

1. With the cursor on the desired prompt, press [Spacebar]

1. Place the mouse cursor to one side of the desired prompt (*not on it*)
2. Drag the mouse cursor across the prompt until it has touched both the prompt label and the data entry area, then release the mouse.

Selecting a group of prompts

1. With the cursor on the first desired prompt, press [Spacebar].
2. Locate the cursor on the next desired prompt and press [Spacebar].
3. Repeat 1 and 2 for any other prompts you want to select.

- If the prompts are located next to each other, drag an outline around them.
- If the prompts are separated so you cannot outline all the ones you want, select one, then press and hold the [Shift] key, then click on each object you want to select. (You will have to click both the label and entry area of a prompt.)

Selecting all objects in a window

Press [Ctrl-U]

Click Edit - Select All.

Selecting labels, lines or boxes

1. Use arrow keys to place the window cursor on the desired object.
2. Press the [Spacebar].

1. Click on a single object you want to select, or
2. Draw an outline around several objects by dragging the mouse.
3. Select additional objects by holding down [Shift] and clicking on the objects.

Deselecting objects

Locate the cursor on the selected object and press [Spacebar].

Click anywhere except on a selected object.

Managing windows

This section outlines tasks that have to do with windows. You will learn how to manipulate the basic characteristics of windows and to set up custom features.

Opening an existing window

1. Choose Define-Window from the Development menu.:
   ( = d w  = **Define-Window**)
2. Paint asks you if you want to open an existing window. Respond "Existing".
3. Enter the name of the table that stores your windows, and the window name, in the collector window. Use [F2] or < Options .> to see a list of existing windows.
4. Save to proceed: ( = [F9]  = < Save>)

Paint will display the specified window and you can begin any modifications. If you need more information about creating, selecting, or editing prompts, labels, lines or boxes, see the topic under "Managing window objects". If you need information about window operations, see "Managing windows". If you need information on adding table columns, see "Managing tables and columns". If you need information on restrictions, validation or formatting, see "Data validation" and "Data formatting".

Resizing a window

Once you've created a window you can change its size. It's easiest with a mouse, but you can use the keyboard also.

1. Choose Resize Window from Paint menu. [Ctrl-F7]
2. Use the arrow keys to drag lower and right-hand borders to the desired locations.
3. Press [Enter] when finished.

If the desired resizing movement is

- VERTICAL: Drag the lower border of the window up or down.
- HORIZONTAL: Drag the right-hand border of the window left or right.
- DIAGONAL: Drag border's lower right corner to the desired location.

Moving a window

You can move the window using the mouse or keyboard. The window must be smaller than the available on-screen space,

1. Choose Move Window from the Paint menu. [Ctrl-F8]
2. Use arrow keys to move the window to the desired location.
3. Press [Enter] when finished.

1. Place the mouse cursor on the window's top border.
2. Drag the window to the desired location, up, down, left, or right.

Saving a window

Use the Save option from the Paint menu or the status line.

( = [F9]  = < Save>)

The first time you save, you can specify a name for the window (unless you opened the window from the Command window). The name will appear as the window title whenever the window is displayed.

Closing a window

When you close a window, you can either go back to the Development menu or remain in the Paint environment and open another window.

- To close the window and leave Paint, select Window-Quit from the Paint menu. (= [F10] w q = **W**indow-**Q**uit)
- To close the window and remain in Paint, select Window-Open from the Paint menu. (= [F10] w o = **W**indow-**O**pen)

Testing a window

If you test run a window that has a table attached, you can write the data you enter directly to it.

1. Choose Test run from the Paint menu. The window displays as an active window. (= [F10] w t = **W**indow-**T**est run)
2. Enter new data, or if a table containing rows exists, press [F2] to select an existing row.
3. Check that the cursor moves through the prompts in the order you desire.
4. If you want to write the test row to the table, press [F9]. Press [Esc] to return to edit mode.

Copying test data to a table

When you begin a test run of a window, Paint will ask you if you want to store data from the test run to the current table. Respond "Yes" if you want to save your test run data.

Copying an existing window

Suppose you need to create a window that is essentially the same as one that already exists, but which presents a slightly different view of the data. Perhaps you want to eliminate one or two prompts, or add some formulas to display totals. In such cases, it is almost always more efficient to copy and modify an existing window than to create a totally new one from scratch.

Note Paint *cannot* copy windows across applications. Advanced Revelation has that capability, but it is outside the scope of this text. Consult the documentation detailing the SET ALIAS command.

To copy an existing window:

1. Open the window you want to copy.
2. Choose Save As from the Paint menu.
   (⌨ = [F10] w a 🖱 = **W**indow-Save **a**s)
3. Enter a window name different from the original.
4. Save the new window name. (⌨ = [F9] 🖱 = < Save >)
5. Make any modifications you want.
6. Save the window in the usual way.

Importing prompts from another window

Paint allows you to import (copy) one or more prompts from another window belonging to the same application. To import prompts from another window, you need to understand several points:

- The window from which you are copying prompts is the **source window**.
- The window into which you are importing prompts is the **destination window**. The destination window *must be currently open.*
- Paint copies the imported prompts *from* the source window *to the Clipboard,* then *from* the Clipboard *to* the destination window.

To import prompts from one window into another:

1. Open the destination window.
2. Display the Import Prompts collector window.
   ⌨ [F10] e i 🖱 **E**dit-Copy **I**mport
3. Enter information about the *source* window (i.e., its Window table and Window name).
4. Press [F9] or click < Save >.
5. Select one or more prompts for import into the destination window from the multiselect popup.
6. Press [F9] or click < Save > to copy your selections to the Paint Clipboard and return to the destination window.
7. Move the window cursor to the location where you want to place an imported prompt.
8. Choose Paste from the Paint menu. ⌨ [F10] e p 🖱 **E**dit-**P**aste
   - If there is only one prompt on the Clipboard, it will paste at the current cursor location.
   - If there are multiple prompts on the Clipboard, a popup allows you to select which one to paste.

9. If you are importing more than one prompt, repeat steps 7 and 8 until you have pasted all the desired prompts into the destination window.
10. Perform any other prompt or window operations in the usual way. You can modify the imported prompts without affecting the source window in any way.

Deleting a window

Before deleting a window, consider:

- Do you really need to delete it? Could you modify it into something else you can use instead of having to delete the window and create a new one?
- Are you certain that deleting the window will not affect any applications other than the one you have in mind? Some systems may use a window in more than one application.

1. Open the window you want to delete.
2. Select Window-Delete from the Paint menu.
   (= [F10] w d = **Window-Delete**)
3. Confirm the delete.

The delete process removes the disk rows of the window and its associated help. You remain in Paint ready to create another window *of the same name.* If you want to do this, proceed to add a table, create prompts, labels, etc., and save the window in the usual way.

If you want to create a new window at this point, select Window-New from the Paint menu. (= [F10] w n = **Window-New**)

If you want to quit paint and lose the window, exit in the normal way and respond "No" when asked if you want to save changes.

Creating multi-page windows

You may find that you have too many prompts to fit them all into one screen. One option is to create a multi-page window, in which there are two or more pages of prompts that you don't initially see. When you cursor to them, however, or if you press [PgUp] or [PgDn], you can jump right to the additional pages—precisely as if they were on attached pages.

To create a multi-page window:

1. Arrange the prompts in each "page" or your window into logical groups.
2. Position the cursor so the first "page" of the window is displayed, with the rest of the pages hidden in in virtual space.

3. Select all the prompts and labels in the page.
4. With all the prompts and labels for that page highlighted, press [F2] or click <Options>. The Prompt Options popup appears.
5. Choose the option Define Page.
6. When you return to Paint, deselect the prompts on that page.
7. Repeat steps 2 through 6 for each page of the window.

For information about navigating a multi-page window, see the chapter "Navigating Advanced Revelation".

Managing tables and columns in Paint

Advanced Revelation provides several methods for creating and modifying data tables. Paint has features that let you create and modify tables without leaving the Paint environment. You can also bind prompts in window to columns in a table so that users running the window can read data from and write data to a table.

Window or table first?

There are two basic ways of approaching the creation of an application:

- You could create the window and then create the table.
- You could create the table first and then create the window.

Neither way is necessarily preferable. The first approach might be more suitable for prototyping situations where you anticipate frequent changes to requirements. You can create windows, etc., show them to users or managers, and then create the data table to support the final window design.

If you know exactly what data elements you need in the table and have a design for the user interface all mapped out, you could take the second approach.

Adding a table to a window

You cannot bind prompts or add columns to a table until you have added the table to the window. Adding the table lets the window update the table or read data from it.

To add a table to a window:

1. Choose the Add Table option. (= [F10] a a = <Add Table>)
2. Select the desired table from the popup.
3. Confirm that the table has been added to the window by checking the status line.

Note Paint automates this procedure when you create a new window. Only a single table may be added to a window.

Adding a new column to a table

Adding a column to the table is part of the prompt binding process, so your window must contain at least one prompt.

From the current window in Paint:

1. Select the data entry area of the prompt.
2. Activate the Editor. Look for "Edit Entry" on the status line.
   (= [F4] = <Options>-Edit)
3. Choose Options. (= [F2] = <Options>)
4. Choose Binding from the popup.
5. Choose Create New Data Column in [*table name*]

Paint creates the new column, and then binds the prompt to the new column.

Note If the prompt name you typed in represents an invalid column name, Paint will ask you to give a valid name for the table column. The prompt name will not change. Paint automates this process when you create a new window.

Binding a prompt to a table column

When creating new prompts as modifications to an existing window, you can either select one prompt at a time, or a group of prompts for binding to the table. The group method is most efficient.

1. Select all of the prompts you want to bind to the table.
2. Choose Prompt Bindings. The Prompt Bindings popup appears listing all the selected prompts. (= [F10] a b = <Options> - Bindings)
3. Select the first prompt you want to bind from the list in the popup.
4. Choose 1-Bind Prompt to an existing column. A popup of available columns appears.
5. Select the column name to which you want to bind the prompt.
6. Select the next prompt in the Prompt Bindings popup.
7. Repeat steps 4, 5, and 6 above to bind other prompts to a table column.

Navigating prompts

Finding a specific prompt

Use Find if you have a window that is larger than the display area of your screen. To use it, you need to know at least part of the name of the prompt you want to find. Note that Find is case sensitive.

To find a specific prompt:

1. Press [Ctrl-F]
2. Type a string you want found (text and numbers OK).
3. Press [Enter]

1. Click Edit - Find
2. Type a string you want found (text and numbers OK).
3. Press [Enter]

Jumping to a specific prompt

1. Make sure no objects are selected in the window.
2. Press [Alt-A]
3. Select the desired prompt name in the popup.

1. Click Edit - Go To Prompt
2. Select the desired prompt name in the popup.

Modifying prompts

Moving a prompt or prompt group

1. Select the desired prompt or a group of prompts
2. Press [F2]
3. Select Drag from the popup.
4. Use the arrow keys to move the prompt.
5. Press [Enter].

1. Select a prompt or prompt group.
2. Drag the selected prompts to the desired location and release the mouse.

Resizing the prompt entry area

1. Select the prompt.
2. Activate the Editor using [F4].
3. Use [←] or [→] to change the size of the entry area.

Deleting a prompt or prompt group

Select the prompt or group of prompts and press the [Del] key. Paint asks you to confirm the delete by pressing [Del] again. The [Esc] key cancels the delete.

Defining the order of cursor flow through prompts

The term **Prompt order** means the order in which the cursor flows through the prompts when the window runs. The system assumes that prompts were created in the order of cursor flow. You can, however, change this default order.

To change the current or default prompt order:

1. Press [Shift-F7].The prompt order popup appears.
2. Move the highlight bar to the prompt you want to move and press [Enter].
3. The prompt name now appears in "Sort Column".
4. Use the arrow keys to move it to the location where you want it to appear.
5. Repeat 2, 3, and 4 for any other prompts you want to move.
6. Press [F9] to store the new prompt order.

1. Select Layout - Reorder Prompts
2. Select a prompt that you want to appear in a different place in the order. (Prompts in the popup appear in the current order of cursor movement)
3. Drag the selected prompt to the desired place in the list.
4. Repeat 2 and 3 for any other prompts you want to move.
5. Click < OK > to store the new prompt order.

Note Test run the window to make sure the cursor moves through the prompts in the correct order. Repeat the above steps and make adjustments if the flow is not right.

Defining the start prompt

The start prompt is the prompt to which the cursor will move after the user enters a value in the key prompt. Normally, you would just create the prompts in the window in the order you want the cursor to flow through them.

To redefine the start prompt:

1. Display the Customize window.
   (= [F10] a u = Advanced-Customize)
2. At the *Start prompt* prompt, enter the prompt *number* (not name) you want to define as the Start prompt.
3. Save your definition. (= [F9] = <Save>)

Modifying prompt details

When you add a new column to a table, you need to define its characteristics or **details**. In doing this, you specify such details as:

- Column name to which a prompt is bound.
- Whether this prompt is a row key or part of a multi-part key
- Data type
- Display and output characteristics and format
- Any desired default value for the prompt

About prompt detail levels

Paint gives you a choice of the level of details you work with when you edit prompt details. You can select **basic** prompt details, or **full** prompt details.
You define prompt details in the Prompt Details window. If you choose the basic detail level, the window contains only a few of the most commonly edited prompt detail options. You can change the level specification any time.

[Shift-F5] sets basic prompt details. Until you specify differently, you will get basic prompt details every time you display the Prompt Details window.

[Shift-F6] sets full prompt details. Until you specify differently, you will get the full array of available prompt detail options every time you display the Prompt Details window.

Displaying the Prompt Details window

1. Press [F2]
2. Select 2-Details from the Prompt Options popup that appears

- Select <Details> from the status line

Set the prompt details to the desired settings in whichever Prompt Details window you are using. Use [F2] or <Options> for guidance on available prompt detail settings. Detailed information for the prompts in the Prompt Details window is available in on-line help.

Changing prompt justification

The default justification for prompts is Left justified. You can change justification to:

Left	The first character of the input is placed in the left-most position of the entry line.
Right	The last character of the input is placed in the right-most position of the entry line.
Center	The input is centered in the entry line.
Text	Input is displayed exactly as entered.
Text Non justified	Any special text formatting characters in the input will be lost when the input is saved. Blanks will be stripped and the text automatically reformatted.

To change prompt justification:

1. Select the desired prompt.
2. Select Prompt Details (basic): (= [F2] 2 [Enter] = <Details>)
3. At the Justification prompt, enter the justification type. Select [F2] < Options> for a list of justification types.
4. Save Prompt detail settings: (= F9 = <Save>)

Converting input to upper case

Prompts accept user input as lower case characters by default. You must specify upper case for all prompts in the window that you want to accept upper case input.

To accept lower case input for a prompt:

1. Select the desired prompt.
2. Display the Prompt Details window. (= [F2] 2 [Enter] = <Details>)
3. A the Lower case prompt enter "N".
4. Save Prompt detail settings. (= [F9] = < Save >)

Creating edit masks for the prompt entry area

An edit mask allows you to specify a character to appear at a prompt before the user enters information. Edit masks show how long the data entry line is and what type of information to enter. For example, you would probably specify *mm dd yy* for a date prompt.

Remember that edit masks only control the display and *don't* perform any type of data validation. Note also that a mask specified for an individual prompt overrides any default mask specified using the Customize window.

Creating a default edit mask

1. Call the Customize window: (⌨ = [F10] a u 🖱 = Advanced - Customize)
2. Enter the mask you want to use as the default in the "Default mask" prompt
3. Save your settings. (⌨ = [F9] 🖱 = <Save>)

Adding an edit mask to a prompt

You can specify a data entry mask for individual prompts that will override any default mask. To specify a default override mask:

1. Select the desired prompt
2. Select full prompt details [Shift-F6]
3. Enter the desired mask characters at the *Edit mask* prompt
4. Save your changes.

Establishing prompts as tab stops

Tab stops allow users to jump to commonly used prompts by pressing [Alt-T].

To place a tab stop on a prompt:

1. Select the prompt
2. Select full prompt details [Shift-F6]
3. At the Tab Stop prompt, select Options and select Yes.

Making prompts required, protected, or filled

Prompt restrictions help developers in their never-ending quest to keep bad data out of the system. Defining prompt restrictions can help you prevent users from entering incorrect data, or failing to enter data at all.

Required prompts

A required prompt prevents the user from exiting the prompt without making a valid entry. The system will display a message if the user tries to skip a prompt that you have defined as required. To define a prompt as required:

1. Select the prompt.
2. Call the full Prompt Details window [Shift-F6].
3. At the *Entry type* prompt, select Options, then 1-"Required" from the Entry type popup.
4. Save the setting.

Filled prompts

Designating a prompt as filled allows you to create a lookup from a table or list and fill the prompt with data from the lookup. Users can change the entry unless you designate the prompt as filled and protected.

Follow the above steps, but select either "Filled" or "Filled and Protected".

Protected prompts

The cursor skips protected prompts to prevent the user from entering data. Use this when you want to display read only data in a prompt.

To define a prompt as protected:

1. Select it.
2. Call full prompt details [Shift-F6].
3. At the *Entry type* prompt, select Options, then 3-"Protected" from the Entry type popup.
4. Save the setting.

Limiting the length or depth of an entry

You can place a limit on the number of characters a user can enter into a prompt, regardless of the length of the prompt entry area. Set limits for prompts where you want the length of the entry limited to the length of the display (or other specific maximum). To place a character entry limit on a prompt:

1. Select the prompt.
2. Call full prompt details [Shift-F6].
3. Move the cursor to the Max. length prompt.
4. Enter the number of characters desired (must be more than zero.)
5. Save the setting.

Limiting number of lines in multivalued prompts

You can control the lines that a multivalued prompt can display. By default, the multivalued prompt will display as many lines as you have allocated for the prompt, but will scroll any number of values through the prompt.

1. Select the prompt.
2. Call full prompt details [Shift-F6].
3. At the *Row limit* prompt, enter the maximum number of lines that should be accepted at this prompt.
4. Save the setting.

About prompt default values

You can specify a default value for a prompt that will display each time the cursor lands in it. The default value can be:

- A literal that you specify
- The same data as entered in the previous row
- System maintained information like date
- A sequential count
- A custom coded default value

Date or time as prompt default values

To have the current date or time appear in a prompt by default:

Current date

1. Select the prompt.
2. Call full prompt details [Shift-F6].

3. At the *Default* prompt, select Options, then 3-"Current date" from the Null Default Options popup. The value %D% appears at the prompt.
4. Save the setting.

Current date and time

Follow the previous procedure, but select 4-"Current date and time" in step 4. The value "%DT%" appears at the *Default* prompt.

Current time

Follow the previous procedure, but select 7-"Current time "in step 4. The value "%T%" appears at teh *Default* prompt.

Last entered value as prompt default

To have a prompt copy the value of the entry from the previous entry, follow the foregoing steps, but select 2-"Duplicate data from previous row" in step 4. A double quote mark (") appears at the *Default* prompt.

Sequential counter as prompt default

To have the prompt default to an incrementing numeric count, follow the preceding steps, but select 5-"Sequence counter (non-key)" or 6-"Sequential key counter" in step 4. For information on the two counter types, see detail on line help in the Null Default options popup.

About labels

Use labels to provide information to users about available keystrokes or other pertinent information. Place a label in any available space in a window. You can convert a label into a prompt, or a prompt into a label. You can also create hidden labels and *smart* labels.

Creating a new label

To create a label, Paint must be in Make Label mode.

1. Select Softkeys-Toggle Prompt-Label.

   (⌨ = [Shift-F2] 🖱 = Bind On/Off)

   Observe the Status Line—it will display the word "Labels" when you enter the Label edit mode:

2. With the cursor in a clear area of the window, type the label text. Label text automatically appears selected, so you can reposition it immediately if necessary. (See "Managing window objects" for more information)

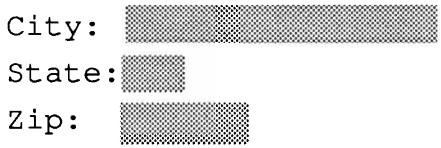
Editing an existing label

1. Select the label and activate the Editor by pressing [F4].
2. Make your changes and press [F4] again to deactivate the Editor.

Hiding a label

Sometimes you might not want the prompt label to appear on the screen with the data-entry part. For example, you might want to display City, State and Zip Code prompts on a single line like this...

...instead of like this:



To define labels as hidden labels:

1. Select the desired prompt
2. Select Options: [F2 <Options>]
3. Select Hide Label: [8 [Enter] Click "Hide Label"]

Note You can undo a hidden label by following the above steps and choosing "Show Label" in step 3.

Creating smart labels

Smart labels are labels that display information. Smart labels can display information such as the system time and date automatically. To define an action for a smart label:

1. Select the Smart Label option from the Label Options popup.
2. Enter a code and command sequence or select Options for a list of available codes.
3. Save your entry when finished. The label will appear at the current cursor position.

For more information on codes and commands, see *Codes and commands*.

About lines and boxes

Paint allows you to draw lines and boxes in windows. Lines and boxes are really a special kind of label and can help you organize prompt groups, call attention to prompts or text labels, etc. You can select from several line styles. The default line style is double line. Drawing lines and boxes is faster using a mouse, but using the keyboard presents no difficulty.

Drawing a line or box

1. Locate the cursor at the location where you want to begin drawing.
2. Press [Ctrl-B].
3. Use the arrow keys to draw the desired object.
4. Press [Enter] when finished. The new object appears selected so you can move it, or delete it if you made a mistake.

1. Click <Options>-Draw Line/Box
2. Move the mouse cursor to the location where you want to begin drawing.
3. Drag the mouse to draw the desired object and release when finished. The new object appears selected so you can move it, or delete it if you made a mistake.

Changing the line style

1. Select Layout-Draw box/line from the Paint menu.
   ( = [F10] l b  = **L**ayout-Draw **b**ox/line)
2. Select the desired line style from the popup display.
   ( = [↑] or [↓] [Enter]  = Click on desired line style)

Customizing windows

Prompts on the Customize window allow you to change how your window looks and behaves as a whole and to define a general window and prompt help structure.

Displaying the Customize window

To display the Customize window, choose the Customize option from the Paint menu.

The first page of the window contains prompts that let you define general characteristics of the window, protect and preserve window information, and add two kinds of window help. The second page contains special registers used for window processing.

Specifying custom window characteristics

The Customize window contains a number of prompts that let you set up custom characteristics for the window.

Border type

You can specify five different styles of window borders. Press [F2] (Options) to display a popup of available border types. The default entry is border type three.

Menu

Names a menu to display instead of the Development menu when the user presses [F10] while running the window. Enter the name of the appropriate menu template stored in the MENUS table.

Status line

Name of the status line to be displayed with the window.

Exploding window

Indicates a special method of displaying a window. The window builds itself from the inside out when displayed on the screen, so it appears to explode.

Start prompt

Indicates the number of the prompt the cursor will move to after the row is displayed. The user must first enter a value in the key prompt. Then the cursor will jump to the Start prompt. This entry line is normally left blank.

Data/Dictionary

This display-only prompt indicates whether the window updates the Data or Dictionary portion of a table. You cannot make a specification at this prompt.

Normally the prompt remains blank, indicating that the window updates a data table. A "D" at the prompt indicates that the window updates a dictionary table.

Default mask

Specifies a character or set of characters to fill in on the entry line for every prompt. Mask characters help the user determine the length of each entry line and what kind of information to enter.

If you enter a mask, the specified characters fill in each time the cursor moves to a new entry line. Make your mask as long as your longest prompt.

You can also specify a mask for an individual prompt with the prompt window. Individual prompt masks override default window masks.

About data validation and formatting

Part of developing applications is protecting the data from entry and formatting errors. Paint provides ways for you to implement data safeguards in windows. Safeguards fall into two basic categories—data validation and data formatting.

Data validation helps prevent users from entering bad data into the system. You can set up checks and processes to sharply reduce the possibility of a user entering data incorrectly.

Data formatting converting data from its optimal storage format into a readable or meaningful display format for windows or reports.

About data validation

About validation patterns

A validation pattern is a defined character format against which the system checks user input data. If it matches, the data is accepted as being good data. If the input data fails the pattern check, the system rejects the entry, displays a message and a list of valid data formats, and allows the user to try again.

Developers frequently set up validation patterns for such data as Social Security numbers, Zip and postal codes, phone or fax numbers, currency types and amounts, and so forth. Advanced Revelation automatically sets up validation patterns for certain data types. When the data entry window updates a foreign file type, like dBASE or

ASCII, the validation pattern ensures that the data entered at the prompt is compatible with the foreign file type.

Types of validation patterns

Advanced Revelation provides several different types of validation patterns. These types are described below.

Literal Matches

Literal matches allow you to check if the data input by a user matches a literal text string. This is useful if you have a limited number of words or other strings that will be acceptable in the prompt.

For example, if you want to establish a pattern match for a prompt that accepts school grades, it might be appropriate to limit the entry to only the letters A through F (less E).

Pattern Matches

Use a pattern match if you want input data to meet a generic format such as "all numeric" or "2 alphabetic characters". The following list shows the available generic patterns (*n* is a number):

Pattern	Meaning
*n*A	*n* alphabetic characters (A-Z, a-z)
*n*N	*n* numeric characters (0-9)
*n*X	exactly *n* number of any character(s)
*n*Z	maximum of *n* characters of any type

(The number zero means "any number of these.")

Examples:

Pattern	Means
3A	"exactly 3 alphabetic characters"
4N	"exactly 4 numeric characters"
5X	"exactly 5 characters of any type"
6Z	"as many as 6 characters of any type"
0N	"any number of numeric characters"

You can use combinations of these to achieve other formats, such as "maximum of 7 numeric characters."

Ranges

You may wish to limit numeric input data to a specific acceptable range. Do this by specifying a starting and ending value for the data.

The syntax for a range check is:

Format	Meaning
(*n*)	All values less than or equal to the number n are valid
(*n*,*m*)	All values between n and m (inclusive) are valid.

For example the entry "(500,9999)" means that the value entered into the prompt must be between 500 and 9,999 (inclusive).

Conversions

Advanced Revelation converts certain types of data into an internal format for storage efficiency and ease of manipulation. These types are:

- dates
- time (e.g., 04:00 AM)
- Boolean values (i.e., Yes/No, True/False)
- decimal numbers (including monetary amounts)

The **conversion** validation pattern allows you to check that input data meets one of these types. The formats of these validation patterns are:

Format	Meaning
(B)	Input data must match the specified **Boolean** string format (default is Y or N)
(D)	Input data must be a valid **date** format
(MT)	Input data must be a valid **time** designation
(DT)	Input data must be a valid **date and time** designation. Either the date or time may be delimited with spaces, but not both.
(MD*n*)	Input data must be a **decimal number** with n decimal places

Advanced Revelation is quite liberal in its interpretation of data to be converted. For example, these are just some of the valid formats in which users can enter a date:

```
01/01/91
JAN 01 1991
01 JAN 91
010191
01-01-91
```

If you specify a conversion validation pattern, the user may enter "Jan 01, 1992", but Advanced Revelation will convert it to a standard format like "01-01-92" *

If you specify the pattern "MD2" to mean that the data must have two decimal places, Advanced Revelation will put in two even if the user enters only one.

Other conversion patterns exist, but are not used frequently. Additional conversions enable you to check for the valid input of hex, octal, or binary data.

Note Conversions check for valid data *and*, if the data passes, *immediately* change the data into internal format. Therefore, you should almost always use conversion validation patterns with data formatting specifications to display the data properly. For more information see "Data Formatting" elsewhere in this chapter.

For more information about input conversion patterns see ICONV in the chapter called "R/BASIC Command Reference" in the *R/BASIC* manual.

Uniqueness validation for multivalued prompts

The **uniqueness** validation pattern lets you to specify that entries in a multivalued prompt must be unique to prevent entry of duplicate values. The validation checks to see if the same data has already been entered at the current prompt in the current row. If the data does not pass the validation, a message informs the user, and he or she can change the entry.

Row exists validation

The Row Exist validation pattern checks the validity of input data by looking it up in a table. The input data must be the key to a row in another table. You can set up Row Exist in either of two ways:

- The input data passes validation if it matches a row in the lookup table.
- The input data passes only if it does *not* already exist as a row in the lookup table.

Row Exist uses the system subroutine *Verifile* to check a specified table to determine if the data matches an existing row key.

Here are some examples of when the Row Exist validation may be useful:

- The data used for validation might change over time.

A sales order application checks the input product code against a Products table, which may be updated frequently.

- There are a large number of possible valid values that remain fairly static.

  The same application might check a Zipcode table for a valid Zip or Postal codes.

- A table contains values you want to exclude from validity.

  The sales order application might check a customer number against a table of customers with poor credit. The input passes only if it does not *exist in the Deadbeat table.*

Establishing a prompt validation

Set up data validation at the *Validation pattern* prompt in the Prompt Details window in Paint. You can set up any number of patterns at this prompt—multiple patterns are treated as either/or validations (see "Combining Validation Patterns" under the "Advanced Data Validations" heading following this section).

To display the Prompt details window:

1. Select the desired prompt.
2. Choose Details: [F2] 2 [ENTER] <Details >

Validating against an exact word or phrase

1. At the *Validation pattern* prompt, enter one or more literal values to match (for example, "ACTIVE", "CLOSED").
2. Save.

Validating against a pattern or mask

1. At the *Validation pattern* prompt, choose Options and select the pattern(s) you want.
2. Modify the patterns to match the exact format that you want to match.
3. Save.

Validating against a numeric range

1. At the *Validation pattern* prompt, choose Options - Range check.
2. Move the cursor and change the *n* and *m* to the desired numbers marking the limits of the range. (Review this topic in the previous section if necessary.)
3. Save.

Validating against date, time, and decimal formats

1. At the the *Data type* prompt., use Options to define date, time, or money (dollar) data types. The system will automatically set up the default date or time validation pattern. If you want a different pattern:
   - Move the cursor to the *Validation pattern* prompt
   - Delete the code that appears in the prompt with [Ctrl-X].
   - Use Options to choose the code for the validation pattern you want (European date, or 24-hour time for example).
2. Save.

Other conversions

Advanced Revelation automatically sets up validation patterns for many of its data types. In most cases you can follow the above steps and select the appropriate data type from the popup. If the data type default validation pattern isn't the one you want, move the cursor to the *Validation pattern* prompt and select what you want using Options.

Refer to "Advanced data validation" if you want to combine the default validation pattern set up by the data type with other validation patterns.

Assuring unique multivalues

1. At the *Validation pattern* prompt., enter the code `"%U%"` (percent signs surrounding a capital U).
2. Save.

Validating against another table ("Row exists")

At the "Validation pattern" prompt, enter in angle brackets (< and >) the name of the table against which an entry at this prompt should be verified. The data entered at the prompt is used as a row key to read a row in the table specified by table name. For example:

```
<CUSTOMERS>
```

makes sure that the entry at the prompt is actually a row key in the CUSOTMERS table.

The full syntax for row-exists validation is:

```
<TABLENAME,{type},{concat.literal}>
```

The argument in the Verifile command must follow the prescribed order, using commas as delimiters. If you skip one of the arguments, enter a comma in the Verifile command line to maintain its position.

The optional type N (not) indicates that the typed information will pass the test if it is not found in the table. If type is null or another character, then the typed information will pass the test if it is found in the table.

Examples:

Verifile syntax:	Meaning:
<PRODUCTS>	Any data entered into the prompt must be the key to a row in the PRODUCTS table.
<PRODUCTS,N>	Any data entered into the prompt must *not* be the key to a row in the PRODUCTS table

The optional concat.literal is concatenated to the front or back of the value typed in the entry line. If the : (colon) is on the front of the literal then the literal is placed at the end of the value. If the : (colon) is in the back of the literal then the literal is placed at the front of the value.

Concatenating validation patterns

Advanced Revelation enables you to string together different validation patterns as required. For example, you can concatenate literals and pattern matches within a single validation pattern.

These validation patterns are typical for phone number prompts:

3N'-'3N'-'4N	Accepts three numerics followed by a dash, followed by three more numerics, followed by another dash, followed by four numerics
('3N') '3N'-'4N	Accepts the same as above, but requires parentheses around the first three numerics, and a space separating the first three from the second three numerics

The validation patterns use both literal text (enclosed in quotes) and pattern matches that specify the number and type of the acceptable characters.

Combining validation patterns

You can also combine different validation patterns when you want the combination of the two to function as a single validation pattern. If you do this, the input data must meet both validation patterns before it is considered valid.

For example, you might wish to create a validation pattern that checks for up to six numeric characters. The pattern match "0N" (any number of numerics) is not adequate, nor is the pattern match "6Z" (up to six of any character). The combination of the two, however, creates the proper validation pattern.

To combine validation patterns into one match, use the underline (_) or plus (+) symbol. For the example above, this would be the format for the validation pattern:

```
0N+6Z
0N_6Z
```

Both mean "any number of numeric characters up to 6."

Range checks can be combined with conversions to put a limit around the values that the conversion will accept. It's important that the conversion be specified as the first part of the pattern match expression. For example, the following pattern combines a conversion for a two-place decimal number with a range check:

```
(MD2)_(500,1000)
```

This validation pattern accepts any number between 5.00 and 10.00, inclusive. In this example, the number 10.001 would pass, since the conversion will truncate any decimal places past the second one.

Note The conversion of input data is done before the range is checked. To check for a valid range of dates, use the internal (integer) format of the date in the range check. There is more information about internal formats in the "Data Formatting" section later in this chapter.

Advanced data validation

Linking multiple patterns with OR

A feature of Advanced Revelation validation patterns is that you can specify several different formats for a single prompt. So long as the data meets any one of the patterns, it is considered valid. To enter multiple patterns with an implied OR, enter each pattern on a new entry line at the prompt. For example, a ZIP code prompt might allow either a 5-digit code or a 9-digit code, since both of these are valid.

Linking multiple patterns with AND

Advanced Revelation also permits you to combine more than one validation pattern into a single check. This enables you to specify that the data must meet several criteria to be valid. To enter multiple patterns with an implied AND, enter each pattern on the same entry line, separating each pattern with a + (plus), subvalue mark, or _ (underscore). For example, you might specify that data is both to be a valid date and to fall within a specified range.

Note When you link multiple patterns with an implied AND, the patterns are processed left-to-right.

Different types of validation patterns can also be strung together ("concatenated"). For example, you can check that an invoice number always begins with the literal "AA" followed by up to 5 numeric characters.

Creating a popup of validation patterns

You can create a popup of the validation patterns you have defined. This popup is called with the [F2] key if no other Options have been defined for the prompt. The validation patterns and the popup that displays them can be defined at the same time. The popup will return a single selection at the prompt. If desired, you can create a display only type popup that does not return a value.

Note Single-selection popups will be most useful at prompts where the validation pattern specification is for literal values. When a single-selection popup is used, the user can select the appropriate value, which is returned at the prompt. Display-only type popups are more appropriate at prompts where the validation pattern specifications are for conversions or pattern matches.

Defining the validation pattern popup

To create a single-selection popup, define the validation patterns normally. Then add a | (vertical bar) to the end of each individual pattern, and follow the bar with a description that is to appear in the popup. The validation pattern that is specified before the | is returned at the prompt.

For example, if your validation pattern dictates that the prompt will accept valid letter grades (A, B, etc.), you can enter the Validation pattern values like this:

```
"A"|Excellent
"B"|Above Average
"C"|Average
"D"|Below Average
"F"|Failing
```

If a user presses the [F2] key at this prompt, a popup appears. The popup displays the letter values in one column and the descriptions in another. The user can choose the appropriate pattern, which is returned at the prompt.

To make the Validation Pattern popup a display only type, add a | (vertical bar) at the end of the last validation pattern description. For example:

```
"A"|Excellent
"B"|Above Average
"C"|Average
"D"|Below Average
"F"|Failing|
```

If a user presses the [F2] or clicks <Options> key at this prompt, and if no other Options have been defined for that prompt, a popup appears. This is a display-type popup, which will not return a value to the prompt.

If a code and command have been specified at Options for this prompt, the Validation Pattern popup will not display.

Data formatting

Advanced Revelation lets you specify formatting characteristics for data. Formatting is useful for data that is stored differently from the way you want to display it. This section discusses how you can format data for display.

About internal storage formats

Advanced Revelation stores certain types of data in an internal format:

- dates
- time
- Boolean values (i.e., Yes/No or True/False)
- decimal numbers, including monetary amounts

The internal format for each of these data types is actually an integer value. The internal format for a date is the number of days before or after December 31, 1967. The internal format for a time is the number of seconds after midnight. The internal format for a Boolean string is 1 (true) or 0 (false). The internal format for a number contains no decimal point or other punctuation.

Advanced Revelation uses internal formats because the data takes up less storage space by excluding punctuation and other extraneous characters. Data access becomes more efficient.

Internal formats also make it easier to manipulate the data. For example, the internal format for dates makes it very simple to calculate elapsed time (as required for aged invoice reports, for example). You never need to account for unequal numbers of days in a month, leap years, etc.

Using internal formats also helps to normalize the data. You don't have to depend on data entry personnel to enter dates or times in a consistent format. Advanced Revelation creates a standard output format based on the internal format of the data.

Internal formats *do* need to be formatted for display, however. The following paragraphs will discuss how to do this. For more details about converting data into internal formats, see the "Data validation" section in this chapter.

Types of formatting

There are two basic types of output formatting available for data: **conversions** and **masking**.

Conversions format data according to one of a number of predefined, generic conversion names. For example, dates are formatted by specifying one of a set of special date conversion names such as "D" or "D2-".

Masks are strings of special characters that represent the actual layout of the display prompt. For example, a display mask for social security numbers might look like this:

```
L###-##-####
```

Where to specify output formats

Specify output formats at the "Output format" prompt in the Prompt Details window. This prompt appears in both the basic prompt details window and the full prompt details window.

1. Select the prompt whose output format you want to specify.
2. Choose Options-Details: [F2] - Details < Options > - Details
3. At the *Output format* prompt, enter the appropriate formatting code(s) or character(s).
4. Save.

The following paragraphs contain information about what you can enter in step 3 above. At that step, you can choose Options again. That will show you a popup from which you can select one of over 30 standard formatting definitions.

Formatting with conversions

To cause Advanced Revelation to format data using a conversion, you simply need to specify the proper conversion name. Conversions are available to format date information, time information, Boolean data, and decimal number or monetary data.

Note For more information on using conversions, see the OCONV entries in the "R/BASIC Command Reference" chapter in the R/BASIC manual.

Dates

Date conversions are flagged with a "D". The format for date conversions is:

```
D year character layout month_format
```

Each attribute is optional. You can use them in any combination, but they must be specified in this order:

- year is the number of digits to display for the year, usually 2 (91) or 4 (1991).
- character is the character that separates the month, day, and year.
- layout determines the arrangement of month, day, and year, for example, mm/dd/yy, dd/mm/yy, yy/mm/dd, etc. The options are:

Option	Format
DE	31 JAN 1992
DF	JAN 1992 31
DG	1992 31 JAN
DH	JAN 31 1992
DI	31 1992 JAN
DJ	1992 JAN 31

- month_format determines how the month should display: as numbers, as an abbreviation, or as a fully spelled out month name. The options are:

Option	Format
none	01/31/1992
S	01/31/1992
M	JAN 31, 1992
L	January 31, 1992

Note Text for the month display depends on the active language set. For more information, see the chapter "International Applications Support" in the appendix of the *Reference* manual.

The following table shows how to use date conversions:

Format Code	Output
D	31 JAN 1992
D2	31 JAN 92
D4	31 JAN 1992
D/	01/31/1992
D2/	01/31/92
D*	01*31*1992
D/E	31/01/1992
D2E	31 JAN 92
D2/E	31/01/92
DS	31 01 1992
D/S	01/31/1992
DM	31 JAN 1992
D,M	JAN 31, 1992
DL	31 January 1992
D,L	January 31, 1992
DJL	1992 January 31
D4.EM	31. JAN 1992
D2.EL	31. January 92

Times

Time conversions are flagged with an "MT". The following table illustrates variations on the display of 2:00PM:

Name	Format	Output Example
MT	HH:MM	14:00
MTH	HH:MMmm	02:00PM
MTS	HH:MM:SS	14:00:00
MTHS	HH:MM:SSmm	02:00:00PM

Date/Times

Date/time conversions are flagged with a "DT". The format for date/time conversions is:

```
DT date_conversion ^ n character time_conversion
```

- *date_conversion* is any valid date conversion.

- ^ separates the date conversion from the time conversion.
- *n* is the number of characters to display between the date and the time. The default number of characters is one.
- *character* is the character that separates the date and time. The default character is a space. To avoid ambiguity, if you specify n, you must also specify character.
- *time_conversion* is any valid time conversion, excluding the "MT" prefix.

For example, to produce a date and time in this format:

```
JAN 31 1992***12:00:00PM
```

use the output pattern:

```
DTH^3*HS
```

Decimal numbers and money

Decimal number and monetary conversions are flagged with an "MD". The general format is:

```
MDnx
```

- *n* is the number of decimal places.

x specifies any character(s) to include in the conversion ('$', commas, etc.)

The following table shows some simple examples of decimal conversions:

Name	Format	Output Example
MD2	nnnn.nn	1234.56
MD4	nnnn.nnnn	1234.5678
MD2,	n,nnn.nn	1,234.56
MD2,$	$n,nnn.nn	$1,234.56
MD2,-	$n,nnn.nn-	$1,234.56-

Note The decimal conversion is exceptionally flexible. For information on the options available, including how to display currency symbols, debit and credit symbols, and null values, see "OCONV Masked Decimal (MD)" in the "R/BASIC Command Reference" chapter in the *R/BASIC* manual.

Formatting with masks

Masks use pound signs (#) to mark a place for each character of the displayed data. These pound signs are combined with other characters (for example, punctuation) and justification information to create a full mask.

A mask always begins with a letter designating how the data is to be justified within the display area created by the mask.

Justification Code	Meaning
L	Left-justify within the display prompt
R	Right-justify within the display prompt
C	Center the data within the display prompt

Rules for Justification Codes:

- Any characters can follow the justification code.
- If characters are pound signs, they will be replaced with data (or spaces).
- Any other non-numeric characters will be treated as literals and embedded into the display prompt.
- When the data from the prompt is displayed, each pound sign in the mask is replaced with a character from the prompt.
- If the prompt is too long for the mask, the display of data is truncated. If the mask is longer than the data in the prompt, the mask will be padded with spaces.

The following table illustrates some sample masks and the output they produce:

Input Data	Mask	Output
AA345	L##-###	AA-345
23	R####	23
345	C#####	345
2066439898	L(###) ###-####	(206) 643-9898
ABCDEFGHI	R#-###-###	C-DEF-GHI

Linking windows to processes and programming

About formulas

A prompt can display the result of a calculation by having a *formula* attached to it. A formula can be any R/BASIC expression. Formula prompts are useful for providing totals, calculating percentages, and other common calculations.

Paint uses Advanced Revelation 's Formula Builder, and hooks a formula created with that tool to a prompt in the current window.

Note In order to create a prompt that uses a formula, the current window must have had a table added to it, and the formula prompt must be bound to a column in the table.

Attaching a formula to a prompt

1. Select the prompt that will use a formula.
2. Display the Dictionary Formula window:
   ⌨ = [F2] 6 [Enter] 🖱 = <Options> - Formula)
3. If the prompt is bound to a data column already, Paint prompts you about converting from a data prompt to a symbolic prompts.
   - Press [Enter]
   - Respond "Yes" to the popup message to convert the table column to Symbolic.
4. Enter the desired formula directly, or press [F2] < Options > to walk through a series of popups that guide you through the various elements of the formula.
5. Save the formula. (⌨ = [F9] 🖱 = <Save>)

Hooking a popup to a prompt

You can have a popup appear almost anywhere in a window to display information or offer you a list of choices. Most often the popup can return one or more selections back to where you called it.

The three most likely places to hook a popup to a prompt are:

- As an option: when the user presses [F2] or clicks on <Options>.
- As detail help: when the user presses [F1].
- As a softkey. This approach is very flexible, because you can then have the popup appear at *any* prompt when the user presses the softkey.

There are other possibilities—at the default value, as part of the validation, etc.—but we'll concentrate on these three options, as they are the most common.

Types of popups

You have several choices for the type of popup that you want to hook to a prompt:

- A popup of all the keys in the current table.
- A popup of all currently-available tables.
- A popup of all locations currently known to your system.
- A custom popup

In all cases, the popups can be either single-selection or multi-selection. The latter is only useful if you hook it to a multivalued prompt.

Before you hook a popup to the prompt, you should:

- Create the popup using the Make Popup window, if you want to use a custom popup. Use the option **Define-Popup** from the Development menu, or choose **Tools-Define popup** from the Paint menu.
- Know the name of the popup. The pre-defined popups (example: all keys in a table) have names like @ID or @IDS (the latter is for a multi-select). If you've created a custom popup, remember what name you've given to the popup. If you have elected to store your custom popup definition in a table other than POPUPS, remember that, too.
- Make sure that the popup is a single-selection popup if you are calling it from a single-valued prompt. If you want to return data to a multivalued prompt, you can define a multi-selection prompt.

Once you have a popup defined, you can then hook it to the prompt in one of the following ways.

Popups as a prompt option

1. Select the prompt.
2. Press [Shift-F6] or click on <Details>.
3. Under the label "Processing" and at "Code" portion of the *Options* prompt, enter "P" (for "popup").
4. At the "Command" portion of the prompt, enter the name of your popup. Press [F2] or click options to see a list of pre-defined popups.

   If you have a custom popup, start with the prefix "POPUPS*", then add the name you've given the popup. If you stored your popup definition in a table other than POPUPS, enter that as part of the prefix instead. The finished entry should look something like this:

   ```
   POPUPS*MYPOP
   ```

5. Save your changes with [F9] or click on <Save>.

Popups as detail help

Use the instructions just above for adding a popup as an option, but instead of hooking it to the *Options* prompt, hook it to the *Detail help* prompt instead.

Popups as a softkey

1. From the Paint menu, choose the option **Advanced-Softkeys**. The Softkeys window appears.

2. At the *Key* prompt, enter the code for the key to which you want to assign the popup. Press [F2] or click on <Options> to see a list of available prompts.

Note Advanced Revelation will not prevent you from assigning more than one function to the same softkey, so be sure your assignment is unique.

3. At the *User description* prompt, enter a short description of what the popup will display. This description appears when the user presses [F6] or clicks on <Softkeys> while in the window.
4. At the *Code* prompt enter "P".
5. At the *Command* prompt, enter the name of your popup. Press [F2] or click <Options> to see a list of pre-defined popups.
6. If you have a custom popup, start with the prefix "POPUPS*", then add the name you've given the popup. If you stored your popup definition in a table other than POPUPS, enter that as part of the prefix instead. The finished entry should look something like this: POPUPS*MYPOP
7. Save your change with [F9] or click on <Save>.

Options for prompt popups

There are two options that may want to add to the code of "P":

Option	Description
P=	Fills in the popup selection and moves to the next prompt without the user pressing [Enter].
P:	For multi-select popups, appends selections onto the end of the current multivalued prompt.

Adding an index lookup to a prompt

If a prompt will contain data that is the key to the current table or to another table, you can use a Btree or Cross reference index to get that data. For example, suppose a prompt is asking for an employee number. If you've put an index on the employee name, you can then use the employee name to look up and return the employee number.

Note If you are at the key prompt and want to use an index for the current table, you don't have to add it as an option—you can run an index lookup using the backslash ('\') key. See the chapter "Using windows for queries".

When you do an index lookup, you provide the indexed data (the name, say), and Advanced Revelation looks up the name in the index. If it finds a single match, it returns the key that matches that data. If there is more than one match, Advanced

Revelation displays a popup of choices from which you can choose the correct match. All of this can be done automatically by specifying an index lookup.

Index lookups are most often added as prompt options so that the user can press [F2] or click on <Options> to start the lookup. However, they can be added to the prompt at any point where you could add a code and command. Alternatives might be a softkey or a detail help key.

Preparing for an index lookup

Before you can add an index lookup to a prompt, you will need to have created an index on the data that you want to use as the lookup. For example, if you want to look up an employee number by the employee name, you will need to already have a Btree or Cross reference index on the employee name. Beyond that you will need to know:

- The name of the table that was indexed.
- The exact name of the column that was indexed. (If you've put a Cross Reference index on the column, the column name you need will have a suffix of _XREF.)
- The names of additional columns that will help you to distinguish the matches if there is more than one. For example, if you look up the name SMITH and get back a dozen matches, it would probably help you if you could also see the first name and department of all those people named Smith.

Defining an index lookup

1. Select the prompt.
2. Press [Shift-F6] or click on <Details>.
3. Under the label "Processing" and at "Code" portion of the *Options* prompt, enter "B".
4. At the "Command" portion of the prompt, enter a three-part specification that has this format:

```
table*indexed_column*display_field1,display_field2
```

"Table" is the name of the table that is indexed, "indexed_column" is the name of the index you want to search, and the "display_field" specifications for columns that should be displayed in the popup if there is more than one match for your lookup. Here is an example:

```
EMPLOYEES*EMPLOYEE_NAME*EMPLOYEE_NAME,FIRST_NAME,DEPT
```

This sets up an index lookup into the table EMPLOYEES, indexed column EMPLOYEE_NAME. If there is more than one hit for the name you enter, a popup displays the columns EMPLOYEE_NAME, FIRST_NAME, and DEPT.

Note Remember that if you want to do a lookup on a column that has a Cross Reference index on it, you need to add '_XREF' onto the name of the index (example: EMPLOYEE_NAME_XREF).

5. Save your changes with [F9] or click on <Save>.

Using the index lookup

1. Press [F2] or click <Options> on the prompt. The Share Index Information window displays with a prompt for the index value.
2. Enter the value you want to look up, or press [F2] or click <Options> to see a list of all indexed values.
3. Press [F9] or click <Save> to start the lookup.

You can use operators (example: greater than, containing, etc.) as part of your search. For more information on searching indexes, see the chapter "Using windows for queries".

Displaying an additional window

You can call any number of additional windows from the current window. You can enter or edit data in those windows, and then return to the current window for further work with the data you have there. Calling windows in this way is useful if you want to update related tables.

Note You can also call "related" windows, which automatically display details about data in the current window. For more information about calling related windows, see "Displaying a Related Window" elsewhere in this chapter.

You can call a window in several different ways: from a softkey, from a prompt option, as detail help, and other places. Here we'll present the first three possibilities.

Before you set up a call to an additional window, you will need to know:

- The name of the window to display.
- If you have elected to store your window definition in a table other than the default WINDOWS table, the name of that table.

You can call another window in the following ways.

As a softkey

1. From the Paint menu, choose the option **Advanced-Softkeys**. The Softkeys window appears.
2. At the *Key* prompt, enter the code for the key that you want to use to call the window. Press [F2] or click on <Options> to see a list of available prompts.

Note Advanced Revelation will not prevent you from assigning more than one function to the same softkey, so be sure your assignment is unique.

3. At the *User description* prompt, enter a short description of what the popup will display. This description appears when the user presses [F6] or clicks on <Softkeys> while in the window.
4. At the *Code* prompt enter "W".
5. At the *Command* prompt, enter the name of the window you want to display.

   If you have stored the window definition in a table other than WINDOWS, prefix your window name with the name of that table surrounded by "@" symbols. For example: @APPWINDOWS@MY_WINDOW
6. Save your change with [F9] or click on <Save>.

As a prompt option

1. Select the prompt.
2. Press [Shift-F6] or click on <Details>.
3. Under the label "Processing" and at "Code" portion of the *Options* prompt, enter "W".
4. At the "Command" portion of the prompt, enter the name of the window you want to display.

   If you have stored the window definition in a table other than WINDOWS, prefix your window name with the name of that table surrounded by "@" symbols. For example: @APPWINDOWS@MY_WINDOW
5. Save your changes with [F9] or click on <Save>.

As detail help

Use the instructions just above for adding a popup as an option, but instead of hooking it to the *Options* prompt, hook it to the *Detail help* prompt instead.

Displaying a related window

If any data in the row you are displaying in the current window is the key to a row in another table, you can use that data to display a related window. When the related window displays, it will contain details about the key data in the current row. If there are several keys in one prompt (at a multivalued prompt), you can even display a browse list of data in the related window.

This is very useful for viewing detail data about the current window. For example, you might be displaying an invoice in the current window. In that invoice you might have a customer number and a list of part numbers. You can use the customer number

and part numbers to display windows in which the full customer details and part details are available for viewing or editing. If the part numbers were multivalued, the related part window would have an active browse list for you to scroll through.

You can set up as many related windows as you like for any window. Before you can set up a related window you must:

- Have a column in the current row that contains keys to rows in another table. These keys don't necessarily have to be displayed in the current window.
- Have a second window that displays data from the table for which you have key data already. For example, if you are displaying an invoice with part numbers, you will need to already have a window for rows in the parts table.
- Know what table you have stored the related window definition in, if you didn't store it in the default WINDOWS table.

To set up a related window:

1. Use Paint to display the source window, the one from which you want to display the second window.
2. Choose the option **Advanced-Relations** from the Paint menu. The Relations window displays.
3. At the *Key* prompt, enter the code for the keystroke that should launch the related window. Press [F2] or click on <Options> to see a list of these codes.

Note Advanced Revelation will not prevent you from assigning more than one function to the same softkey, so be sure your assignment is unique.

4. At the *Source table* prompt, enter the name of the table in which you've stored the definition for the related window. If you've stored your windows in the WINDOWS table, you can skip this prompt.
5. At the *Related Window* prompt, enter the name of the related window. Press [F2] or click on <Options> to see a list of windows available.
6. At the *Joined column* prompt, enter the name of the column in the current row that contains the key or keys to rows in the related table. Press [F2] or click on <Options> to see a list of columns in the current table.
7. Save your change with [F9] or click on <Save>.

To display the related window from the current window, simply press the keystroke that you identified in step 3. To display a list of all related windows, press [Ctrl-F6].

Displaying a menu

You can call a menu from the current window. You can enter or edit data in the windows called by the menu, and then return to the current window for further work with the data you have there. Calling menus in this way is useful if you want to update unrelated tables.

You can call a menu in several different ways: from a softkey, from a prompt option, as detail help, and other places. Here we'll present the first three possibilities.

Before you set up a call to an menu, you will need to know:

- The name of the menu you want to call.
- If you have elected to store your window definition in a table other than the default WINDOWS table, the name of that table.

Calling a menu as a softkey

1. From the Paint menu, choose the option **Advanced-Softkeys**. The Softkeys window appears.
2. At the *Key* prompt, enter the code for the key that you want to use to call the menu. Press [F2] or click on <Options> to see a list of available prompts.

   Note Advanced Revelation will not prevent you from assigning more than one function to the same softkey, so be sure your assignment is unique.

3. At the *User description* prompt, enter a short description of what menu will display. This description appears when the user presses [F6] or clicks on <Softkeys> while in the window.
4. At the *Code* prompt enter "M".
5. At the *Command* prompt, enter the name of the menu you want to display.

   If you have stored the menu definition in a table other than MENUS, prefix your menu name with the name of that table surrounded by "@" symbols.
   For example: `@UTILMENU@MY_MENU@`
6. Save your change with [F9] or click on <Save>.

Calling a menu as a prompt option

1. Select the prompt.
2. Press [Shift-F6] or click on <Details>.
3. Under the label "Processing" and at "Code" portion of the *Options* prompt, enter "M".
4. At the "Command" portion of the prompt, enter the name of the menu you want to display.

   If you have stored the menu definition in a table other than MENUS, prefix your menu name with the name of that table surrounded by "@" symbols. For example:

   ```
   @UTILMENU@MY_MENU@
   ```
5. Save your changes with [F9] or click on <Save>.

Calling a menu as detail help

Use the foregoing instructions for calling a menu as a prompt option, but instead of hooking the command to the *Options* prompt, hook it to the *Detail help* prompt.

Pre-prompt processing

You can specify processing that is to take place before a prompt displays on screen. For example, you might want to require a password before allowing changes to some data. For more information, see "About Processing and Windows" in the Managing Windows section.

Post-prompt processing

You can specify processing that is to take place immediately after as user enters data in a prompt. For example, if a window is recording invoice payments, you could compare the prompts PMT_AMOUNT and BALANCE_DUE. You could then specify what processing takes place next, based of the results of the comparison. For more information, see "About processing and windows" in the Managing Windows section.

Creating custom softkeys

You can define custom softkeys for your windows. Your users will see and select them in the softkeys popups.

Note Defining custom softkeys involves the use of Advanced Revelation's code and command sequences. Review the chapter "Codes and commands" if necessary.

Assign keys in logical function groups. For example, you might use Shift+key combinations for all softkeys that display windows, and Alt-key combinations for those that print reports.

To define custom softkeys:

1. Display the Softkeys window from the Paint menu.
   [F10] a t Advanced-Softkeys
2. Specify which key to use for the custom softkey you are defining:
   a) Enter the key name or select from the listing of available keys by choosing Options.
   b) Press [Enter].
3. Type a brief description of the action the softkey will perform and press [Enter].
4. Specify the code for the action you want the custom softkey to perform:

a) Type the code or select from the listing of available codes by choosing Options.

b) Press [Enter].

5. Enter the desired command and press [Enter].
6. Repeat 2, 3, 4 and 5 for as many softkeys as you want to define (maximum 20).
7. Save your softkey definitions. ⌨ [F9] 🖱 <Save>

Disabling keys in a window

You can prevent users from performing certain operations by disabling specific keys. For example, you might disable the [F9] key in a display-only window to prevent updates to the table. To disable keys:

1. Choose Key Disable from the Paint menu:
   (⌨ [F10 a k] 🖱 Advanced-Key disable)
2. Select the keys you want to disable from the selection popup that appears. [Enter] toggles the select on and off for each item in the popup.
3. Save your choices: (⌨ [F9] 🖱 < OK >)

About Help for windows

You can create two basic levels of on line help in your applications—detail help for prompts, and concept help for windows. For information about window concept help, see "Managing Windows".

Advanced Revelation provides a number of possible ways for creating help in applications. To understand all the possibilities, you need a good working knowledge of Advanced Revelation's codes & commands. For the sake of simplicity, we will show you two methods which even novice Advanced Revelation developers can use to create help while working in Paint.

Creating detail help

Before beginning to create help for prompts in a window, there are two things you must do first:

- Save the window containing the prompts
- Give the window a name

This is because Paint looks for and updates certain system tables when you begin to create and save help. Paint needs the window name to do its job.

To create help for a prompt:

1. Be certain you have saved the window and given it a name.
2. Test run the window. (⌨ [F10] w t 🖱 Window-Test run)
3. Place the cursor at the prompt for which you want to create help.
4. Press [F1]. A message will display indicating that no help is available, and asking if you want to create your own help.
5. Respond "Yes" to the message. This will call the editor.
6. Type your help text in the window. All the Editor's navigation keys work in the usual way.
7. When you finish writing help text for the current prompt, press [F9] or click <Save> to save your work to the help for the current window.
8. Press [Esc] to exit from the Editor.
9. Repeat steps 3-8 for all other prompts for which you want to write help text.
10. Press [Esc] to quit the test run.

Note Save the window again before you quit, or you risk losing your help text.

Creating window (concept) help

Window or concept help provides information to the user about how to use the window as a whole. We recommend that you create detail (prompt-level) help before creating concept help for the window. For information on creating detail or prompt-level help, see "Creating detail Help" just above.

Creating help involves the use of Advanced Revelation's codes and commands, so you need be familiar with them. Review the chapter "Codes and commands" if necessary.

To create concept help from within Paint using codes and commands:

1. Display the Customize window from the Paint menu.
   (⌨ [F10] a c 🖱 Advanced-Customize)
2. Move the window cursor to the Concept code prompt.
3. Enter a concept code or choose Options for a list of available help codes and sample commands.

Recalculating symbolic prompts

As an application window executes, it will normally recalculate all symbolic fields whenever the user enters data at a prompt. The window will do this for every symbolic field each time it runs, whether changes affected a symbolic field or not.

In most cases, not every symbolic field in the window needs to be recalculated for every prompt entry. Avoiding unnecessary recalculation helps speed processing. In Paint you can specify when to recalculate symbolic fields, choosing from several recalculating schemes and directing the window processor to use the method that best suits your window design.

To specify window recalculation:

1. Select **Advanced-Recalculations** from the Paint menu.
   (⌨ [F10] a e 🖱 Advanced-Recalculations)
2. Specify the type of recalculation in the Recalculate popup.

Each selection on the Recalculate popup recalculates the window's symbolic fields slightly differently. The following paragraphs describe the different schemes.

All: Immediately recalculates all symbolic prompts in the window each time values at other prompts change. This is the default setting for a window. For example, you might create an Invoice window that contains a multivalued Item Amount prompt used to enter individual line item amounts and a symbolic Invoice Total prompt that calculates the total. By selecting "All", you direct the window processor to recalculate the invoice total after each item amount is entered.

None: Recalculates the value of any symbolic prompt in the window when the cursor moves to that prompt. If you specified "None" for the example Invoice window described above, the invoice total would be recalculated only when the cursor moves to the Invoice Total prompt. Specifying "None" can save time in a window where each change might otherwise set off lengthy calculations.

Custom: Lets you specify which symbolic fields to recalculate at what time. This option displays a list of all the prompts with symbolic fields for this window. Press [Enter] to toggle the message displayed beside each prompt from "only as used" (None) to "every change" (All) as desired.

Dependency: Directs the window processor to calculate only the symbolic fields that are dependent upon the prompt entry that is currently being edited. This option provides top window processing speed while ensuring that the symbolic field data displayed in the window is accurate and up-to-date. Symbolic fields that are not dependent on data entered at other prompts in this window will be calculated and recalculated only when initially displayed or when they are accessed in the window's prompting order.

Security for windows

Establishing security is an important aspect of developers' efforts to preserve data integrity. There are three classes of security that you can implement in windows:

Application	You can specify in which application(s) the window will run.
Security Level Restriction	You can assign a minimum security level to a window. Users whose security level does not meet the window's minimum restriction level cannot access the window.
Password	You can assign one or more passwords to a window. Users whose security level would normally bar them from a window can access it if they know at least one of the assigned passwords.

To implement security in windows, you need to be familiar with the ways Advanced Revelation provides for security. Review the section on security in the chapter "Creating and managing applications and users".

Defining window security

1. Display the Window Security window.
   ([F10] a s Advanced-Security)
2. Specify the appropriate security information in each of the three security categories.

About row locking

The locking options indicates the type of locking you want for rows displayed in the window. Locking protects information when the more than one user accesses the database. If your application will run on a network, be sure to use row locking to protect your data.

The five locking options are described below.

Note These locking options are only functional on networks that support these kinds of locks. See your network manual to determine which types of locking are available to you.

To select a window locking option:

1. Display the Window security window as described previously.
2. Select [F2] or < Options > to display a popup list of locking options.
3. Choose an option by positioning the cursor and pressing [Enter].

Row lock options

Exclusive lock: (Default locking method) The first user who accesses a row obtains exclusive write privileges for that row. Other users can display an exclusive locked row, but they cannot update it. A locked out user sees messages indicating that the row

is in use, that the displayed data is protected. The locked out user must access the row when it is no longer being used by someone else.

No locking: The window can be used to access and update any row in the table. When a read or write occurs, nothing checks to see if any other process is using the row or has attempted to lock it. No locking allows access to, and overwriting of a locked row.

Caution! Specifying the no locking option can have a serious impact on data integrity in a network environment. Do not specify it unless you are certain that the application will never run on a network.

Shared lock: Use with "view only" windows where the window itself cannot update rows. Shared locking in these windows prevents users with table update capabilities from updating a row while someone else is viewing it. You can make a window "view only" by disabling the [F9] (Save) key or substituting another process for the write process.

Exclusive coordinated lock: Like an exclusive lock, except it checks to see if the table itself is locked before locking the row. If the table is locked, the row will not be locked and vice-versa. Use this option in windows when some process in the application may lock all rows in the table.

Note A process that places a lock on all rows in a table only recognizes coordinated locks. You must use an exclusive coordinated lock whenever the window updates a table that is sometimes used in a process that employs table locking. This is the only way to ensure that you can obtain an exclusive lock on the row when using the window. Users locked out by exclusive coordinated locking can only display the locked row . Processes that lock the entire table cannot be run until the lock is released.

Shared coordinated lock: Use in "view only" windows to prevent a process from locking the table while the shared lock on the current row is in effect. Shared locking prevents an exclusive coordinated lock. No process that uses exclusive coordinated locks can update the table while a shared coordinated lock is in effect on the table.

Note A process that places a lock on all rows in a data table only recognizes coordinated locks. You must use shared coordinated locking whenever the "view only" window accesses a data table that is sometimes used in a process that employs table locking. This is the only way to ensure that the current row remains unchanged during the time that it is displayed in the window. Users locked out by this method can only display the locked row. Processes that lock the entire table or require an exclusive lock cannot be run until the shared coordinated lock is released.

Joining prompt columns with other tables

Windows can display and update information from more than one table at a time. In addition to the table assigned to a window, you can specify columns in different tables that will be updated by entries made at the individual prompts in the window. A join column writes to a column in a row in another table.

Joins let you store information for a window in several different tables, allowing you to create an efficient database structure.

For example, within a Customer window that updates the CUSTOMER table, you might include a Notes prompt that updates a separate table used exclusively for storing customer notes. You might specify NOTES as the new table name, CUSTOMER_ID as the key, and "1" as the column's position number at the Join prompts.

You establish a link between the column associated with a prompt and a column in another table using the Join prompts that appear on the second page of the Prompt window. While in the Prompt window with the Editor off, you can press [PgDn] to display the Join prompts.

Table: Specifies the name of the additional table to be joined to this window. This table will be updated with data entered at the prompt. Press [F2] (Options) to see a list of all attached tables.

Key: Specifies the name of the column in the current row that contains the key to the row in the additional table. This key is used to join the other table.

> Example: The Customer window described above contains a CUSTOMER ID prompt. If you used a separate table for customer notes, you would specify CUSTOMER_ID as the key column so that every note you write would be associated with a particular customer.

Position: Indicates the position number of the column in the other table that will store the information entered at this prompt. To determine the position, look the column up in the dictionary of the joined table. If the other table contains only one column, as a NOTES table might, you would specify "1" for the position number.

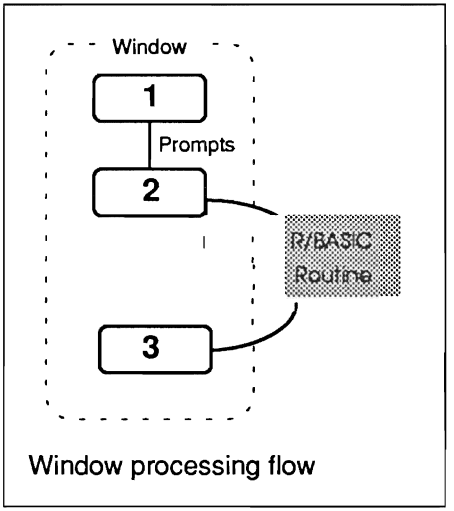
Delete Join Row: Indicates if the joined row from the other table is to be deleted automatically when the associated row in this table is deleted. Again consider the NOTES table example. If you specified "yes" at the Delete join row prompt, whenever you deleted a customer row, the associated notes row would be deleted automatically.

Changed Key Delete: Indicates whether to delete the associated row in a joined table if the value entered in the key column in the current table changes. You would enter "no" if you want to preserve the associated row in the joined table even though the value in the key column changes.

About window processing

Special processing on windows and prompts allows your application to process information automatically as it runs. Processing can be added to the window flow as needed with code and command sequences.

When a window runs, Advanced Revelation lets you call R/BASIC code at selected points to perform special operations. When the process is finished, control returns to the window.



Window processing flow

To use window processing, you need to understand what system variables are available at any given point in the window flow and what effect your code will have on those variables.

Defining window processing

To define processing for a window, choose the Processes option from the Paint menu. This displays the Process window.

The Process window allows you to add special processing to normal window operation, giving you complete control of the flow of data through a window. Code and command sequences are used for this purpose and can be called from any prompt in the Process window.

A Code and command can be executed from this window, the most commonly used code is S (subroutine). A subroutine called from a prompt in the Process window allows you to access or modify the WINDOW_COMMON% variables, a group of COMMON variables initialized in a window and used to store window, prompt and data information.

Note Advanced developers can find additional information about using the WINDOW_COMMON% variables in the "Programming with R/BASIC" chapter in the R/BASIC manual.

Each prompt in this window is described below in the order in which it is executed. Descriptions of the Pre-Prompt and Post-Prompt processes are also included.

Note All I/O processing in windows is standardized. If you want to prevent further processing from a Pre process, set the WINDOW_COMMON% variable WC_RESET% to 5 (RESET_RECORD). Replace and Post processes will not execute. If you want to stop processing from a Replace process, set the WC_RESET% variable to a 5. The Post process will be ignored, if it exists.

Pre-Application: Processes information before the window template is read. You might use a pre-application process to specify Table mode for a window and call an introductory help message. A pre-application process is specified on the command line when you start the window. For more information, see the WINDOW entry in the "TCL" section of the Reference.

Pre-Initialize: Processes information after the template row is read but before it is divided into variables and displayed on the screen. A process here might check to see what account this window is called from and remove several unnecessary prompts.

Post-Initialize: Processes information after the template row has been assigned to variables but before it is displayed on the screen.

Pre-Read: Processes information after the user has typed a row key and pressed [Enter]. It allows you to perform a process before the specified row is read. For example, you might have contractors and permanent staff on the same personnel list. You could check to see if that row is for a contractor and, if so, display a message window for contractors.

Replace-Read: Replaces the standard read procedure used by Advanced Revelation with your own read process. For example, you may want to use a special operating system read process that accesses parts of a row larger than 64K.

Post-Read: Processes information after the row is read, but not yet displayed. For example, you might have three fields called 30, 60, and 90 in a Receivables table that are used to display a message indicating when a customer is overdue. Your process could check a value in a particular field to determine whether to display one of these messages.

Pre-Prompt: Processes information before the user starts typing information in a prompt. For example, you might do some special processing to see if you should skip this prompt. Pre-prompt processing is available for every prompt in the application window.

Post-Prompt: Performs a process based on what has just been entered in the prompt. For example, you might call an R/BASIC subroutine in a phone number prompt that

adds an area code to the number automatically if none was entered. Post-prompt processing is available for every prompt in the application window.

Invalid: Performs a process in response to data that does not pass the In Pattern check. For example, if you typed in an invalid product type, the process could ask you if you want to add that to the valid product types.

Perpetual Process: Performs a process between every prompt in a window. This process occurs immediately after the Post-Prompt process. For example, you might compare a payment amount with an invoice amount and, for all prompts where the payment is less than the invoice amount, change the display to the color red.

Pre-Save: Processes information after the user presses [F9] but before the row has disappeared from the screen. For example, you might require that a minimum set of information be entered in a customer window (company, contact, and phone) before permitting a Save.

Replace Save: Replaces the standard save procedure used by Advanced Revelation. For example, you could use a save process that writes to a specific device instead of a table.

Post-Save: Processes information after the row is written. For example, you might use a process to read each product key from a multivalued field in a product table. You could then subtract the number of products from the total quantity field to keep a running total of the quantity on hand for any item.

Pre-Refresh: Processes information after the user presses [F8] but before the window is refreshed. A process might verify that a user is modifying an existing row and then prevent refreshing the window if a new row is being entered.

Replace-Refresh: Replaces the standard refresh process used by Advanced Revelation. A Replace-Refresh process could identify data to be displayed for the next row in the window.

Post-Refresh: Processes information after a window is refreshed. For example, you might lock rows in another table during a process that is run from your window. Using a Post-Refresh process, you would unlock these rows if the user refreshes the screen.

Pre-Delete: Processes information after the user presses [Alt-D] but before the row is permanently removed. For example, you could write an invoice row to an invoice archive table before it is removed permanently. This would systematically archive rows.

Replace-Delete: Replaces the standard delete processes used by Advanced Revelation. For example, you could flag a row for deletion, and later use a process to actually delete the row.

Post-Delete: Processes information after a row is deleted but before the row key is removed from memory. For example, you could concatenate the row key and write it to a register. A post-application process could then save a list of all the keys written to that register.

Post-Application: Processes information when the user exits the window. For example, you could write a key list of new or changed rows to the LISTS table.

9. Creating menus

About menus

Menus give users access to application windows and reports without having to learn the Advanced Revelation commands and menu system. By choosing options from menus, users can run application windows, print reports, and perform a variety of other tasks.

You can create different menus for different users. In a payroll application, for example, you may want to give a data entry person and an accountant different sets of options. If you're concerned about security, you can restrict access to specific menu selections or to entire menus.

Planning a menu system

Design the menus for your application after you have set up the application windows and reports. That way you can be sure to include a menu option for all the application parts.

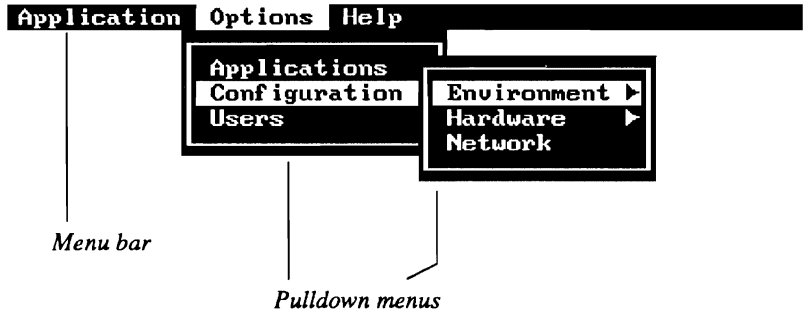
Group your menu options together according to order and frequency of use. For example, you might group all window options on a main menu and all the report options on a report menu accessible from the main menu. You might also provide a way to exit the application from the report menu.

Menus can call other menus. For larger applications, you may want to use a number of different menus all accessible from one main menu.

Menu bars and pulldown menus

Menus come in two types, menu bars and pulldown menus. An menu bar is the menu that displays on the top line of the screen, and a pulldown menu is one that displays below a selection in the menu bar.

- If your application has only one menu, that menu automatically displays as an menu bar.
- If you have more than one menu, the main menu automatically displays as an menu bar. The menus that are displayed by the main menu automatically display as pulldown menus. If selections in pulldown menus display other menus, the other menus also display as pulldown menus.



Anchored menus

One of the characteristics of a menu is whether the menu redisplays when an operation is complete. For example, if you have a reports menu, do you want that menu to redisplay when a report is finished running, or would you rather display the menu bar?

If you identify a menu as anchored, the menu will redisplay when an operation is complete. If the menu is not anchored, the last anchored menu displays. If you're four pulldown menus down from the menu bar and none of the intermediate menus is anchored, you'll return to the menu bar when an operation is complete.

Using codes and commands

Each selection on a menu can perform one of over a dozen different operations, including:

- Display a window.
- Display another menu.
- Display a help message.
- Run a TCL command.
- Run an R/BASIC program.

For each selection on a menu, you must specify the type of operation that you want that selection to perform (for example, display a window) and the specifics associated with that operation (for example, the window name). The type of operation is identified by a code and the specifics by a command. The following table provides some examples:

Task	Code	Command
Display the RECEIVABLES window	W	RECEIVABLES
Display the REPORTS menu	M	REPORTS
List rows in the EMPLOYEES table using the TCL LIST command	E	LIST EMPLOYEES
Display the CAR_PARTS popup	P	POPUPS*CAR_PARTS
Display the help message in the HELP table that has a row key of RECEIVABLES_REPORT	HF	RECEIVABLES_REPORT

For more information on codes and commands, see the chapter "Codes and commands" later in this book.

Creating a menu

When you create a menu, you're creating an entry in a table. This entry, known as a template, contains all of the information that the system needs to display and run the menu. Unless you specify otherwise, all menu templates are stored in the MENUS table.

Before you create a menu, you should know:

- The name of each item on the menu (menu selection). The names should be short and consistent. For example, on a Reports menu, every selection should end with the word "Reports" or none should (Receivables report, Payables report, etc.).
- The operation that you want each menu selection to perform, for example, display a window or run a report.
- The particulars corresponding to that operation, for example, the window name or the report name.

To create a menu:

1. Choose the option **Define-Menu** from the Development menu. The Make Menu window displays.
2. At the *Table name* prompt, press [Enter] to save the menu template in the MENUS table, or enter the name of the table that you want to save the menu template in.
3. At the *Menu name* prompt, enter a name for the menu. For list of menus already defined, press [F2] or click <Options>.
4. In the *Selection* column, enter the name of a menu selection as you want it to appear on the menu.

5. In the *Description* column, enter the long description of the menu selection, up to 63 characters. This description appears in the status line at the bottom of the screen when the selection is highlighted.
6. In the *Code* column, enter the code for the type of operation that you want to perform. Press [F2] or click <Options> for a list of valid codes.
7. In the *Command* column, enter the detail information related to the code.
8. Repeat steps 4 through 7 until you've entered all of the selections for this menu. Enter each menu selection as a separate row.
9. At the *Anchored (Y/N)* prompt, enter "Y" if you want this to be an anchored menu, otherwise enter "N".
10. Press [F9] or click <Save> to save the menu template.

Testing a menu

You can test a menu to see whether it works as intended. While displaying your menu template in the Make Menu window:

- To test a menu as an menu bar, press [Shift-F10].
- To test a menu as a pulldown menu, press [Shift-F1].

Adding help to a menu

You can add three types of help to a menu, as indicated in the table below:

Type of help	Press	Comments
Concept help	[Ctrl-F2]	Use concept help to describe the overall purpose of the menu, for example, to run accounts receivable reports.
Detail help	[F1]	Use detail help to describe the individual selections on the menu. You can have a separate description for each selection on the menu.
Default help	[F1]	Use default help if you want to use the same description for all selections on the menu.

Adding help to a menu is a two-step process. You must add a row to the HELP table that contains the help text. In addition, you must add the name of this row to the menu template so that, when a user presses [F1] or [Ctrl-F2], the appropriate help text displays. You can perform these steps in either order, but if you want to test the help, the row must already be in the HELP table.

Adding concept help

To add concept help for a menu:

1. Display the Make Menu window and then display the menu template that you want to add help to.
2. At the *Concept help* prompt, enter the name of the row in the HELP table that contains the help text for this menu.
3. Press [F9] or click <Save> to save your changes.

Adding detail help for menu selections

To add detail help for a menu selection:

1. Display the Make Menu window and then display the menu template that you want to add help to.
2. Move the cursor to the selection that you want to add help to.
3. Press [Shift-F4] to display the Item Detail window.
4. At the *Detail help* prompt, enter the name of the row in the HELP table that contains the help text for this selection.
5. Press [F9] or click <Save> to save your changes.

Adding default help

To add default help for a menu:

1. Display the Make Menu window and then display the menu template that you want to add help to.
2. At the *Default help* prompt, enter the name of a row in the HELP table that contains the proper help text.
3. Press [F9] or click <Save> to save your changes.

Adding rows to the HELP table

To add rows to the HELP table:

1. Press [F5] to display the Command window.
2. Enter:

```
EDIT HELP row_name
```

where *row_name* is the name that you entered at one of the help prompts in the Make Menu window.

3. Enter the text that you want to display when a user presses [F1] or [Ctrl-F2] at a menu.
4. Press [F9] or click <Save> to save your changes.

Additional menu options

Changing the hotkey for a menu selection

Menus work with "hotkeys". When you create a menu, the system automatically picks a character in each selection to be the hotkey. It does this by choosing the first unique character in each selection (but defaulting to the first character of an item when all available letters have been used).

To override the hotkey chosen by the system, place an ampersand ("&") immediately before the character to be highlighted. For example:

```
In&vestigations
```

highlights the "v" in the menu item Investigations.

To include the ampersand in a menu item (for example, in "Fun & Games"), double the ampersand:

```
Fun && Games
```

Reordering menu selections

To reorder menu selections:

1. Display the Make Menu window and then display the menu template that you want to reorder.
2. Press [Shift-F2] to display the Menu Order popup.
3. Highlight the selection that you want to move and press [Enter]. The selection name appears in the right column.
4. Move the highlight to where you want the menu selection to appear, and press [Enter].
5. Press [F9] or click <Save> to save your changes.

Alphabetizing menu selections

To alphabetize the selections in a menu by selection, press [Shift-F3].

About menu security

You can restrict access to a menu or to individual menu selection by specifying a restriction level, a password, or both.

- If you specify only a restriction level for a menu or menu selection, a user must have a restriction level equal to or lower than the menu restriction level. Otherwise, the menu or menu selection won't display. For example, if the menu restriction level is 3 and a user's restriction level is 4, the menu won't display for that user.
- If you specify only a password, the system prompts all users for the password before displaying the menu.
- If you specify a restriction level and a password, only the users whose restriction level is higher than the menu restriction level are prompted for a password. For example, if the menu restriction level is three and your restriction level is 2, the menu displays for you without further ado. If your restriction level is 4, the system prompts you for a password before displaying the menu.

If menu item access is so restrictive that a user can't see any of the selections on a menu, the menu displays "(Empty)".

You can avoid maintaining separate passwords for users and for menus (or menu selections). Just specify "??" for the menu password, and users can enter the password they entered at logon to satisfy menu security. This also saves users the trouble of remembering separate logon and menu passwords.

Caution! Do not use the "??" password if users can log in without entering a password. A user with no logon password can access every menu in the system that has a password of "??"

Passwords are saved in encrypted form. This prevents a user from checking a menu template to find out what the password is for that template.

You can also restrict access to a menu to specific applications. If you don't indicate which applications have access to a menu, the menu is accessible to all applications.

Adding a restriction level or password to a menu

To add a restriction level or a password to a menu:

1. Display the Make Menu window and then display the menu template that you want to add security to.
2. Press [Shift-F5] to display the Menu Security window.
3. At the *Restriction level* and *Password* prompts, enter a restriction level, a password, or both. You can enter more than one password. Enter each password on a separate line.
4. Press [F9] or click <Save> to save your changes.

Adding a restriction level or password to a menu selection

To add a restriction level or a password to a menu selection:

1. Display the Make Menu window and then display the menu template that you want to add security to.
2. Move the cursor to the menu selection that you want to add security to.
3. Press [Shift-F4] to display the Item Detail window.
4. At the *Restriction level* and *Password* prompts, enter a restriction level, a password, or both. You can enter more than one password. Enter each password on a separate line.
5. Press [F9] or click <Save> to save your changes.

Restricting menu access by application

To restrict the applications that can access a menu:

1. Display the Make Menu window and then display the menu template that you want to add security to.
2. Press [Shift-F5] to display the Menu Security window.
3. At the *Applications* prompt, enter the names of the applications that you want to have access to this menu. You can enter more than one application. Enter each application on a separate line.
4. Press [F9] or click <Save> to save your changes.

Displaying a main menu in an application

You can specify the menu that displays when a user logs into an application. For more information, see the chapter "Creating and managing applications and users" earlier in this book.

10. Creating popups

About popups

Popups display information to the user in table form. In some popups, the user chooses one or more rows of information to use in further processing. These popups can return the value that the user selects to the location the popup was called from, for example, to a prompt in an application window. In other popups, information is just displayed. In either case, the displayed information can be drawn from a table or can be explicitly entered as part of the popup definition.

Once you define a popup, you can call it from anywhere in your application. Typical examples include attaching a popup to the [F2] Options key for prompt in a window, as a softkey, or as a menu option.

Planning for popups

Examine your application and try to anticipate places where users may need additional information. For example, you might have a popup available from a Contact Name prompt that displays the names of all the contacts that work for a particular company. Or, at a Product Code prompt, you might supply a popup that lists codes and descriptions to remind a user what products are normally carried.

Single- and multiple-selection popups

If you are not familiar with the different types of popup, please read the section "About popups" in the chapter "Navigating Advanced Revelation".

The data that a popup displays

A popup can display data from any of three sources:

- Literal values that you enter
- Rows and columns from a table

Display literal data

If you enter literal values, those values appear in the popup just as you entered them, in the order that you entered them.

Displaying data from a table

If you display data from a table, you specify the columns to display. The system uses the following priority to determine which rows to display:

- The rows that you specify by row key.
- The rows represented by an active select list. Use a select list if:
  - You only want to display some of the rows in a table. You can use selection criteria to limit the row keys that will appear in the select list and, therefore, in the popup.
  - You want to display the data in an order other than row key order. You can use sort criteria to sort the row keys in a select list and, therefore, in the popup.
- Every row in the table, in row key order, if the table has a Quickdex or Rightdex index.
- Every row in the table in no particular order.

For more information on select lists, see the chapter "Using windows for queries". For more information on Quickdex and Rightdex indexes, see the chapter "Creating and maintaining indexes".

The data that a popup returns

You can create a popup that is for display only. However, in general, popups are used to help the user by displaying a list of values that the user can choose from. A common example is a popup that returns a value to a prompt in a data entry window.

For a popup that returns data, the data returned can be:

- One or more values that appear in the popup. A popup can display several columns of information but can only return values out of one column, although a multiselect popup can return several different values out of that column.
- The row keys of the rows that you choose from the popup.
- The numbers of the rows that you choose from the popup. For example, in a single-select popup that contains 20 rows, if you choose the 19th row, the popup returns the value 19.

Deciding where to display the popup

You can position a popup anywhere on the screen. When deciding where to display the popup, consider the size and location of the popup in relation to other windows on the screen. For example, if an application window appears in the top left corner of the screen, you may want to position the popup in the top right corner of the screen so a user can still see the window when the popup is displayed.

Converting data for display (output format)

Advanced Revelation saves several types of data, including dates, times, currency, and decimal numbers, in an internal format that doesn't look much like the display format. For example, the time 8:28 p.m. is saved as 73680. If you want to display a column whose data is saved in one of these special formats, you must specify a conversion so the data appears in a recognizable form. For more information on conversions, see the chapter "Creating windows with Paint".

Creating a popup

To create a popup that displays values that you enter in the Make Popup window, that displays rows from a table, or that displays values from a row, follow the appropriate instructions below.

Creating a popup that contains literal values

To create a popup that contains literal values that you enter at the Make Popup window:

1. Choose the option **Define-Popup** from the Development menu. The Make Popup window displays.
2. At the *Table name* prompt, press [Enter] to save the menu template in the POPUPS table, or enter the name of the table that you want to save the popup template in. For a list of available tables popups, press [F2] or click <Options>.
3. At the *Popup name* prompt, enter a name for the popup.
4. At the *Column* and *Row* prompts, enter the coordinates for the upper left corner of the popup.
5. At the *Selection process* prompt, enter a "0" (zero) if you want this to be a single-selection popup. Enter a "1" if you want this to be a multiple-selection popup.
6. At the *Mode* prompt, enter "R".
7. Leave the *Source table* prompt blank.
8. Enter the text for the popup at the *Explicit popup information* prompt. Enter each row on a separate line. Separate columns with a "|". The following example creates a popup two rows long and three columns wide:

```
KP|Kurt Prince|Kaiser & Prince
LT|Loni Taylor|Aupro Corporation
```

9. Page down. At the *Title* prompt, enter a title for the popup. The title can be more than one line.

10. In the *Column formats* section, enter values to define the format of each column of the popup. For each column of information that you specified at the *Explicit popup information* prompt (as separated by "|"), you should have a separate line here. In the example above, you have defined three columns, so you need three lines of entries under "Column formats":
    - *Position* is the column number. The leftmost literal data is column 1, the data after the first "|" is column 2, etc.
    - *Width* is the width of the column in characters.
    - *Just* is the justification of the data within the column: L (left justified), R (right justified), or C (centered).
    - *Conversion* is the data conversion (output format). If the data to display is a date, time, currency, decimal number, or other data requiring a conversion or output format, enter a conversion pattern to convert data from Advanced Revelation internal format to the appropriate display format. Press [F2] or click <Options> to see a list of common output formats.
    - *Column heading* is the text that appears above this column. Don't use the "|" character in a column heading.
11. If you want to be able to press [F1] and display help at this popup, at the *Popup detail help key*, enter the row key of a row in the HELP table.
12. At the *Column to return when selected* prompt, enter the column number of the column whose value should be returned when the user selects something from the popup. For example, if you wanted the popup to return the company name in the above example, you would enter "3" here.
13. If you want to return a value that doesn't appear in the popup or if you want the popup to be display only, enter a value at the *Type of information* prompt: P (return the number of the row *in the popup*), or R (return no value—display only). The return type overrides any entry that you make at the *Column to return* prompt.
14. Save your definition with [F9] or by clicking <Save>.

Creating a popup that contains data from a table

A table popup lists rows out of the table. Each row of the popup displays one data row from the table. Each column of the popup displays a data column from the data row. You cannot display multivalued column data in a popup.

Before you create a popup that contains data from a table, you must know whether any of the columns that you want to display are saved in a special internal format. If so, you'll need to specify a conversion (output format) for these columns so the data display in a recognizable format.

A note about "Columns" in the Make Popup window

Note that when you are creating a table-mode popup, the word "Column" in a prompt name means different things at different times in the Make Popup window. Because this can be quite confusing, you should understand these differences.

When referring to the position *to display*, "Column" means the position that a column represents in the data rows being displayed in the popup. For example, the key column is always position zero in a data row, so if you want to display keys in the popup, you should indicate that you want to *display* column zero.

However, when referring to a column *to return*, "Column" means the column as it appears in the popup. For example, if the row key appears in the third column in the popup, and that's the data you want to return from the popup, you would ask to *return* column 3 (rather than 0, which is the position that the key occupies in the data row).

To create a popup that contains data from a table:

1. Perform steps 1 through 5 of "Creating a popup that contains literal values", directly above.
2. At the *Mode* prompt, enter "T" for "table".
3. At the *Source table* prompt, enter the name of the table from which you want to display values.
4. At the *Explicit popup information* prompt, enter:
   - Nothing, if you want to display all the data rows in the table, or a select list of data rows.
   - The keys of the data rows, if you want to display an explicit list of data rows from the table. If the row key is a multipart key, you must specify the key parts in order. Separate key parts with an asterisk ("*")
5. Page down. At the *Title* prompt enter a title for the popup. The title can be more than one line.
6. In the *Column formats* section, enter values to define the format of each column of the popup. For each column in the table that you want to display in the popup, you should have a separate row here.
   - At *Position,* enter the name or the *row position* (as listed in the dictionary) of the column to display. For example, if you want to display the company name column, enter either the column name (e.g. CO_NAME) or the position that the company name column represents in the data row. Press [F2] or click <Options> to see a list of columns defined in the dictionary of the table you are displaying. If you select a column name, the column position will be filled in at the prompt.
   - At *Width* enter the display width for the column in the popup.

- *Just* is the justification of the data within the column: L (left justified), R (right justified), or C (centered).
- *Conversion* is the data conversion. If necessary, enter a conversion pattern to convert data from Advanced Revelation internal format to the appropriate display format.
- *Column heading* is the text that appears above this column. Don't use the "|" character in a column heading.

7. If you want the user to see help when pressing [F1], at the *Popup detail help key*, enter the row key of a row in the HELP table.
8. At the *Column to return when selected* prompt, enter the column number *in the popup* whose value should be returned when you leave the popup. For example, if the company name displays as the second column in the popup (regardless of its position within the data row), enter "2" here.
9. If you want to return a value that doesn't appear in the popup or if you want the popup to be display only, enter a value at the *Type of information* prompt: "K" (return the row key), P (return the number of the row *in the popup*), or R (return no value-display only). The return type overrides any entry that you make at the *Column to return* prompt.
10. Save your popup definition with [F9] or by clicking <Save>.

Testing a popup

To preview your popup while you are defining it, press [Shift-F1]. Advanced Revelation displays the popup as it would appear according to your current definition.

Displaying popups with codes and commands

You can display a popup anywhere that a code and command sequence is permitted. Popups are commonly defined as options on individual application window prompts.

To display a popup, use the P code. The general format is:

Code	Command
P	table_name*popup_name

table_name is the name of the table that popup definition is saved in, and *popup_name* is the name of the popup that you want to display. Separate the table name and row name with an asterisk (*).

For more information about the use of codes and commands, see the chapter "Codes and commands" in this book.

11. Creating messages

About messages

Use messages to ask for information from the application users, report the progress of a process, and report error messages.

You can create your own messages or use system messages.

Parts of a message

The major components of a message are:

- The message number, the key to the row in a message table.
- The message type.
- The message text.

About message numbers

Message numbers are keys to rows in a messages table. Message numbers:

- Can be up to 50 characters long (but should be short enough to fit into the message border).
- Cannot contain spaces.
- Should start with an alphabetic or numeric character.
- Should be mnemonic. A message number should provide some indication of where the message is being called. For example, if a message is called from the Sample_Invoices window, you could use the numbering convention INVWIN01.

Message types

For more information, see the system subroutine, MSG, in "R/BASIC Reference" in *R/BASIC*.

There are four basic types of messages:

- Timed messages appear for a specified period of time.
- Acknowledged messages display until a key is pressed or the <OK> button is clicked.
- Response messages ask for information or user input.
- Progress messages report the status of the current process.

Creating messages

Create messages in the Message Builder window. Messages you create in this window are stored in the MESSAGES table.

To create a message:

1. From the Development menu, choose **Define-Message.**
2. At the *Message Id* prompt, enter the key to the row in the MESSAGES table.
3. At the *Type* prompt, press [F2] to display the Message Types popup and select the message type you want.
4. Enter the script to appear in the message at the *Text* prompt.
5. Press [F9] or click <Save> to save the message.

Using N type messages

N type messages are often used to suppress the display of system messages.

Note If you want to revise a system message, copy it from the SYSMESSAGES table to the MESSAGES table. Then modify the row in the MESSAGES table.

Using progress type messages

When you create messages which will post process status messages to the screen, you first display a UB (up border) type message with text. The message stays posted while your process runs. When the process has completed, display a DB (down border) type message which takes down the up border message and returns the screen to its pre-process appearance.

Entering the message text

For more information, see "Changing the appearance of the message" later in this chapter.

The *Text* prompt in the Messages window is a multivalued prompt. If the message is defined to be text justified, each line you enter at this prompt will appear as a separate paragraph in the message. With right, left and centered justification, each line will appear as one line in the message and will be truncated if necessary to fit within the message borders.

Adding replaceable parameters in a message

You can include replaceable parameters in message text. These are filled when the message displays. The most common parameters are:

Parameter	Meaning
%D%	current date
%T%	current time
%B%	beeps (rings the bell)
%C*n*%	ASCII character *n*

Testing the message

To test run a message, press [Shift-F1]. When you test response-type messages, test run will display the response you entered.

Message options

To modify default message characteristics, press [Shift-F2] to display the Advanced Message Options window. Press [F9] or click <Save> to save your entries in this window.

Changing the appearance of the message

In the Advanced Message Options window, you can specify:

- The location of the message at the *Column* and *Row* prompts. By default, messages are displayed in the center of the screen.
- One of five border types at the *Border Type* prompt.

- Left, right, center or text justification of the message text at the *Justification* prompt. For left, right and center justification, each line you enter at the *Text* prompt will appear as one line in the message. For text justification, each line you enter at the *Text* prompt will appear as a paragraph in the message.
- The width of the message at the *Width* prompt. For left, right and center justification, the text will be truncated if it is longer than the width specified at the *Width* prompt.

Validating responses

For response-type messages, you validate or convert user responses by specifying a validation pattern at the *Response validation* prompt. Press [F2] or click <Options> to see available validation options. For information on creating validation patterns, see the chapter "Creating windows with Paint".

For more information, see "Codes and commands" in the *User's Guide.*

Specify the action to be taken if the response fails the validation test at the *Invalid Code* and *Invalid Command* prompts. For example, to display a custom error message, enter the code HL (a literal help message) at the *Invalid Code* prompt and at the *Invalid Command* prompt, enter:

```
Please enter alphabetic characters.
```

Documenting the message

Use two of the prompts in the Advanced Message Options window to enter general information about the message. At the *Programs* prompt, enter the R/BASIC programs which use the message you are creating. At the *Category* prompt, enter the purpose for the message. You might want to enter "E" for an error message and "P" for a progress message.

The information at these prompts is for your use—Advanced Revelation will not make use of the values entered here. You can use the data in these two columns to select specific messages and then edit or create reports for documentation.

Displaying messages

For more information, see the system subroutine MSG in "R/BASIC Reference" in *R/BASIC*.

You can use the codes and commands HM or HMD to display a message. Use the HM code to display a help message and request a user response. Use the HMD code to display an acknowledged type message. For example, to display a message when a user presses [Shift-F1] in a window, enter an HM or HMD at the *Code* prompt in the Softkeys window, and the number of the message at the *Command* prompt. You can also display a message by calling it from within an R/BASIC program using the MSG subroutine.

Editing system messages

When Advanced Revelation displays a message, the system first searches the MESSAGES table and then the SYSMESSAGES table for the message. Therefore, you can modify system messages by copying them to the MESSAGES table and modifying them there.

Note We recommend that you do *not* modify messages in the SYSMESSAGES tables, as your changes can be overwritten by a subsequent upgrade.

To edit system messages, follow these steps:

1. Copy the message in the SYSMESSAGES table you want to modify to the MESSAGES table. Choose the **DB Admin-Rows-Copy** option from the Development menu to use the Copyrow window to copy the message.
2. Change the message in the Messages window.
3. Save the message.

12. Using codes and commands

About codes and commands

Using codes and commands, you can:

- Display popups, windows, menus, and messages.
- Access indexes.
- Display various types of help.
- Run TCL commands.
- Run R/BASIC programs.

Use codes and commands to link together parts of your application. Code and command work together as a shorthand method for accessing Advanced Revelation tools. The code specifies the type of operation to perform and the command specifies the operation.

Where to use codes and commands

For more information about using code and command, see CATALYST in "R/BASIC Reference" in the *R/BASIC* manual.

You can use codes and commands almost anywhere when defining windows, forms, menus, macros, applications, users, and keystroke capture sequences. You can also use codes and commands to add custom default and validation processing to window prompts, or even call them within an R/BASIC program.

Codes and commands for index lookups

B

Use the B code when you want to use an index to look up the row key for a known data value. If more than one value is found, a popup displays. The syntax of the command is:

```
table*indexed_column*display_column1,display_column2*
   mode
```

"Table" is the name of the table that is indexed, "indexed_column" is the name of the index you want to search, the "display_column" specifications for columns that should be displayed in the popup if there is more than one match for your lookup and "mode" specifies what will occur if more than one key is returned.

Mode	Action
KEY	The indexed rows are displayed in a multiselection popup and the returned row keys are placed in a browse list.
MULTI	The indexed rows are displayed in a multiselection popup and the returned row keys are ready for display in a multivalued column.
Null	The row keys are displayed in a single-selection popup.

Here is an example:

```
SAMPLE_CUSTOMERS*COMPANY_NAME*COMPANY_NAME,CITY
```

This sets up an index lookup into the table SAMPLE_CUSTOMERS, indexed column COMPANY_NAME. If there is more than one hit for the company name you enter, a single-selection popup displays the columns COMPANY_NAME, CITY.

V

Display a popup of data values from an indexed column. The syntax for the command is:

```
table,column,code
```

Table is the name of the indexed table, and column is the name of the indexed column. The code can be M or S. M displays a multiple-selection popup and S displays a single-selection popup. For example, a code of V and a command of CUSTOMER,CUSTOMER_TYPE,S will display a single-selection popup of all customer types on table.

Code and command to display popups

P

Use the P code to display a popup. The command is required. The command can specify a system popup or a popup row.

Displaying a system popup

To specify an available system popup, use one of these commands:

If you want	Enter
A single-selection popup of row keys	@ID
A multi-selection popup of row keys	@IDS
A single-selection popup of attached tables	@FILE
A multi-selection popup of attached tables	@FILES
A single-selection popup of attached volumes	@VOLUME
A multi-selection popup of attached volumes	@VOLUMES

Displaying a custom popup

To specify a popup row defined in the Make Popup window, use the syntax:

```
table_name*row_name
```

Table_name is the table where the popup template is located and row_name is the key to the popup template. For example, the code P and the command POPUPS*MEMBER_TYPES displays the popup template MEMBER_TYPES from the POPUPS table when it is activated.

Code and command to display menus

M

Use the M code to display a menu. The command is required and contains the name of the menu to display. For example, the code M and the command APPMAIN displays the Development menu.

Code and command to display windows

W

Display a window using the W code. The command is optional and contains the window name. When the command is a window name, that window displays. If no window name is specified, the Window Name window displays.

Window command options

When you want to	Use	Example command
Display a window where the template is not stored in the WINDOWS table	@table_name@	@SAMPLE_TEMPLATES@ SAMPLE_CUSTOMERS
Display a window with a browse list active	row key list	SAMPLE_CUSTOMERS 10 12 14
Display the table browser	TABLE.ON	SAMPLE_CUSTOMERS TABLE.ON
Display the window with window query active	QUERY.MODE	SAMPLE_CUSTOMERS QUERY.MODE
Use another data table with the same dictionary layout.	DATAFILE = table_name	SAMPLE_CUSTOMERS DATAFILE=CUSTOMER

Codes and commands to display help messages

Use the H codes to display help. Each type of H code provides a different way for the information to display. The syntax for the command is determined by the type of H code used. The available codes are:

H

Use the H code to display help stored in the SYSHELP or the HELP table. The command is a number that is placed at the end of the row key. The row key has this format:

```
AW*template_name*number
```

Template_name is the name of the window using this help and the number is commonly the prompt number for the help. For example, the code H and the command 1 when used for detail help will display the help for the first prompt in the window where it is used.

HD

Use the HD code to display help from the Description column in a dictionary row. The command is optional. If the command is not filled, the help information is read using the current prompt name as the key to the dictionary row. If a command is used, it contains the name of the dictionary row with the help information. For example, the code HD and the command ST will display the help from the dictionary row ST.

HF

Use the HF code to display help from a row or the contents of a column in a row in any specified table. This makes it possible to use a table other than HELP to store your help rows. The command is required and is typed using one of three syntaxes:

```
table_name, key, column_number
table_name, key
key
```

Table_name is the name of an attached table, key is the name of the help row, and column_number is the number of the column for the help information. For example, the code HF and the command

```
HELP,CHECK_REGISTER, 4
```

displays the help information in the fourth column of the CHECK_REGISTER row in the HELP table.

The code HF and the command

```
PRE_PRINT
```

displays the help information in the PRE_PRINT row in the HELP table.

HL

Use the HL code to display a help message that is typed at the command prompt. The command is required and contains the help message. For example, the code HL and the command "Type a five digit, nine digit, or Canadian postal code." will display the typed message.

HM

Use the HM code to display a help message and request a user response. The message is stored in the MESSAGES or SYSMESSAGES table or the message text is whatever you specify at the command prompt. The command is required and is typed using one of these syntaxes:

```
key
literal_message_text
```

The response is returned in the system variable @ANS. A response can be returned to the prompt from which the message was called.

HMD

Use the HMD code to display an acknowledged type message. The command is required. It contains the key to a row in the MESSAGES or SYSMESSAGES table or it contains the message that is displayed.

HT

Use the HT code to display a timed help message. The command is required and contains the message to be displayed. The default duration of display for timed messages is set in the Message Environment window. To explicitly specify the number of seconds that the message should display, enter ~ (a tilde) followed by the appropriate number. For example, the code HT and the command "Make sure the printer is on and ready.~10" will display the message for 10 seconds.

Codes and commands to run TCL commands

E

For more information, see EXECUTE in the "R/BASIC Reference" chapter in *R/BASIC.*

Use the E code when you want to execute a TCL command. The command is required and may be any command used at the Command window. Enter the command using the same syntax as at the Command window. No active row key list is passed to the command. For example, the code E and the command EDIT BP BORDER edits the program BORDER in the BP table when activated.

X

For more information, see PERFORM in the "R/BASIC Reference" chapter in *R/BASIC.*

Use the X code when you want to perform a TCL command that uses an active list of row keys. The command specified may be any command used at the Command window. Enter the command using the same syntax as at the Command window. An active list of row keys can be passed to the command, or the command can generate an active key list to be passed back to the calling routine. For example, the code X and the command LIST CUSTOMER lists the CUSTOMER table when performed. Any active list of row keys will be used by the LIST command.

Code and command to use a symbolic column

For more information on using dictionary rows, see { }, and CALCULATE in the "R/BASIC Reference" chapter in *R/BASIC.*

{

Use the { code when you want to use a symbolic column in the current dictionary. The command is required and is the name of a column in the current table. The { code will run the formula in the specified column and return the result. For example, the code { and the command INV_TOTAL returns the value of the INV_TOTAL column.

Codes and commands to run R/BASIC programs

C

For more information, see "Programming with R/BASIC" in *R/BASIC.*

Use C to compile and run small R/BASIC programs. The command is required and is the R/BASIC source code. Each line of code can be separated by an ampersand (&). For example, the code C and the command @ANS = SUM({AMOUNT}) sums the column AMOUNT and returns the total.

Caution! Every time the C code is run, the R/BASIC program in the command will be compiled and then executed. Therefore, to avoid slowing your system and using memory, use the C code sparingly and limit the size of the R/BASIC code.

S

For more information on calling subroutines, see CALL in the "R/BASIC Reference" chapter in *R/BASIC.*

Call a cataloged R/BASIC subroutine using an S code. The command is required and is the name of a subroutine or a subroutine with an argument. The command syntax is:

```
subroutine [,argument]
```

The subroutine must be cataloged. The optional argument must be a value, a literal, or the number of a window register. For example, the code S and the command AR_SUB calls the AR_SUB program when the code and command are activated. The command AR_SUB, MONTH_END calls the AR_SUB program and passes it a single argument, MONTH_END.

The S code also supports the use of <n> (where n is the number of a register) to pass the contents of a window register. The command AR_SUB,<4> calls the AR_SUB program and passes to it the contents of window register 4.

Codes and commands to run keystrokes

@

Use the @ code when you want to execute a captured keystroke set. The command is required and is the name of the capture set to execute. The capture set is an entry in the CAPTURED table. The syntax for the code and command is:

```
@ capture_set_name
```

F

Use the F code when you want to execute a function associated with a function key such as [F1] or [Alt-F1]. The command uses a code to indicate which function key to operate. The following are examples of key codes:

- F1 for [F1]
- SF1 for [Shift F1]
- CF1 for [Ctrl F1]
- AF1 for [Alt F1]

For example, the code F and the command F2 when used as a pre-prompt process will activate the [F2] <Options> key before placing the cursor at the entry line of the prompt.

K

Use the K code when you want to execute a series of keys. The command uses a code to indicate which keys to operate. The command is filled by pressing [Ctrl - –] (Control + hyphen), which toggles the keystroke translator, then typing the keystrokes required to execute the desired process. When you have finished typing the keystrokes press [Ctrl - –] to toggle the keystroke translator back off. There is a pause before the first keystroke is converted and printed to the command entry line. For example, the code K and the command {CTRL-F3} {CTRL-U} when the editor is on will cut a block to the active buffer and clear the block definition.

Miscellaneous codes and commands

For more information, "Creating windows using Paint" in the *User's Guide*.

!

Use the ! code when you want to display the date, time and date, or the Advanced Revelation serial number. The command can be the words DATE, TIMEDATE, or SERIAL. This allows you to display this information as a "smart" label in a window.

The code and command options

This section lists options that can be used in conjunction with any code and command.

Calling multiple codes and commands

Use ; (semicolon) to execute a series of codes and commands consecutively. The semicolon follows each code and each command to be executed. For example,

```
S;S;P PROGRAM1;PROGRAM2;POPUPS*INSURANCE_CODES
```

runs three codes and commands in succession. First the subroutine PROGRAM1 executes, followed by PROGRAM2. Finally, the INSURANCE_CODES popup displays.

=

Use = (equal sign) to generate an automatic carriage return. Use the = if a single value is returned to a single-valued prompt. For example,

```
B= SAMPLE_CUSTOMERS*COMPANY_NAME*COMPANY_NAME,CITY
```

will return one selection to a prompt and then move the cursor to the next prompt.

:

Use the : (semicolon) to append any return value onto the end of the data already at a multivalued prompt. For example,

```
B: SAMPLE_CUSTOMERS*COMPANY_NAME*COMPANY_NAME, CITY
```

will append the returned selection to the data already displayed at a multivalued prompt.

#

Use # (hash mark) to protect help. Append # to the code. Entering H#, for example, will not allow help displayed from the HELP table to be edited by the user.

P

Use the P option with any code to suspend the playback of captured keystrokes before executing the code and command. When the code and command process is completed, playback resumes.

Codes and commands are added to a keystroke set by editing the keystroke row in the CAPTURED table. The code and command should be on its own separate line, and should have the format @code command@. For example,

```
@SP SIMULATE_INPUT@
```

could suspend keystroke playback and run a subroutine called SIMULATE_INPUT that requires the user to enter keystrokes. When the user responds appropriately, execution of the code and command is complete. At that point, keystroke playback will resume.

R

Use the R option with any code to save away the screen image before that code is performed. At the completion of the code and command, the screen image is restored. For example, you might use the R option to restore the screen display following the execution of a code and command that prints to the screen:

```
SR INVOICE_RPT,MONTH_END
```

X

Use the X option with any code to specify that a post-prompt process code and command will run only if the data at the prompt changes. For example, the X option might be used in conjunction with an S code and command to run a post prompt process that writes an entry to an audit table each time the value at a prompt changes:

```
SX AUDIT_UPDATE
```

Querying the database

13. Introduction to queries in Advanced Revelation

About Advanced Revelation reports

This chapter introduces the Advanced Revelation tools used to produce reports. It also explains how to customize the data dictionary for reporting purposes and how to use the View window to examine reports.

Advanced Revelation offers you four kinds of reports: forms, labels, lists, and merge reports (reports that combine and format text with varying quantities of data). Each type of report can be predefined for use in an application.

For example, a sales application might require reports that produce invoice forms, customer mailing labels, a phone list, and sales letters personalized for each customer. Each of these reports would print information from a particular table. Invoices would be printed using data from the Invoices table, while the other reports might draw their data from the Customer table.

- Forms are used to lay out your columns to match paper forms, such as insurance forms, and print data to fill in the blanks. Successive forms on a page print in rows across or columns down the page. You can print one or more rows per page.
- Labels are used to create mailing labels. You specify the number of lines for the label and how many labels to print across a page. Labels are also convenient for producing address lists for reference or for editing.
- Lists are often used when you want to see the contents of a table in columns and rows. Lists can also show relationships between rows by displaying totals or by grouping rows into categories.
- Merge reports merge data from an Advanced Revelation table into a form or document that also includes text, variable information such as date and time, headings and footings, and character fonts such as boldface and italic.

The tools used to design a report are listed on the **Define Report** menu. When you design a report, you create a template for the report and save it in the REPORTS table. After creating the format, you can print the report through the **Run** menu or from the Command window.

Customizing the data dictionary for reports

The data dictionary tells Advanced Revelation what to do with the data that is stored in a column. When Advanced Revelation prints an R/LIST or Label report, it uses information in the data dictionary to determine the display width and justification of each column. You can adjust these settings using the Dictionary window. In addition, if you want custom column headings printed in an R/LIST report, you can create the headings in the Dictionary window.

This section explains how to use prompts in the Dictionary window to specify the column width and justification that will display in R/LIST and Label reports and to customize column headings in the report.

Displaying the data dictionary

To display a data dictionary, choose **DB Admin-Columns-Define** from the Development menu. The Dictionary window appears.

Setting output format

Use the *Output format* prompt to control the data's output pattern. You can tell Advanced Revelation to display the data in a specific format, regardless of how the data was entered. For example, you can specify that Advanced Revelation is to display a date in the format mm-dd-yy. Regardless of what format was used to enter the date, Advanced Revelation displays the date in the pattern you set.

For more information on output formats, see "Data Formatting" in the chapter "Creating windows with Paint".

Adjusting column justification and length

Use the *Justification* and *Display length* prompts to control the column's justification and the length of the output lines.

Justification

By default, Advanced Revelation displays data in a column with left justification. To change the justification of a column, enter the appropriate option at the *Justification* prompt. To see all the options, press [F2] or click <Options>.

For example, if all of the values in a column are a single digit, you may want the values centered under the column heading. In that case, enter C at the prompt.

Display length

By default, Advanced Revelation allows ten spaces for the length of a column when the column is printed. To change the length, enter the appropriate number at the *Display length* prompt.

Customizing a column heading

By default, Advanced Revelation uses the column name as a column heading on R/LIST reports. To change the heading, type the heading you want at the *Column heading* prompt.

Using View for R/LIST reports

Advanced Revelation's View option displays R/LIST reports in a special View window. Reports in this window can be paged forward and backward, and reports wider than the screen can be panned side-to-side. This permits you to scroll through an entire report in both directions. In addition, you can display reports wider than the screen without causing lines to wrap to the next line.

Displaying a report using View

When View is active, any R/LIST report you create will be displayed in a special View window. R/LIST reports can be generated from the Command window using LIST or SORT, from a special R/LIST processor such as EasyWriter, or directly from a window using [Alt-V]. For more information on creating reports using R/LIST, see the chapter "Using EasyWriter" in this section. Also see "Introduction to R/LIST" in the *Reference* book..

If the report is narrower than the screen, the View window will be correspondingly smaller. If the report is wider than the screen, the window will show a dashed border, with the hidden portion of the report off the dashed edge of the screen. This hidden portion can be seen by panning in the appropriate direction.

View has no effect on reports directed to a printer.

Keys used in View

When a report is displayed in a View window, special keys enable you to pan forward and backward and side-to-side in your report. Other keystrokes change the number of columns shifted when you pan, permit you to jump to a specific page of the report, enable you to modify your report, or route your report to the printer.

Summary of Keystrokes Active in View

Type	Key	Meaning
Movement Keys	[Enter], [↓], [PgDn]	Display next page of View Report
	[↑], [PgUp]	Display previous page of View report
	[Ctrl-PgUp]	Display first page of report
	[Ctrl-PgDn]	Display last page printed so far
	[Home]	Pan report to first column
	[End]	Pan report to last column
	[←]	Pan to show additional columns left
	[Ctrl-←]	Pan left one window width
	[→]	Pan to show additional columns right
	[Ctrl-→]	Pan right one window width
Special Keys	[Spacebar]	Increment number of columns shifted while panning left-right
	[Backspace]	Decrement number of columns shifted while panning left-right
	[Ctrl-G]	Go to a specific page of the report printed so far
	[Alt-V]	Modify and restart report
	[Alt-P]	Print the current report on the printer from the beginning

Modifying and restarting the View report [Alt-V]

To modify and re-execute the View display while in a View window, press [Alt-V]. View will display the current R/LIST sentence in an R/LIST window.

You can change the R/LIST sentence displayed in the R/LIST window by editing it. When you press [F9] or click <Save> at the window, the current report will be regenerated from the beginning of the report with the changes you have made in the R/LIST window.

Note If your original report was running from a list of row keys (a select list), the select list will not apply to your modified report. For example, if you had used GETLIST to activate a select list before running your original report, that select list will no longer be active for the modified report.

Routing a View report to the printer

If you press [Alt-P] while in a View window, the current report will be routed to the printer. The report will be regenerated, and then printed from beginning to end without pausing.

While the report is printing, you cannot pan the report, nor can you use [Alt-V] to modify the display of the report. However, once the printed report is finished, you will be returned to the place in your View window from which you pressed [Alt-P], and can continue viewing the report.

14. Using windows for queries

Selecting and browsing through data

This chapter provides an overview of how Advanced Revelation uses lists of row keys to locate the rows used in a program or command. It also explains how row key lists are used in an application window to view specific rows from a data table.

Using row keys

If you are not already familiar with row keys in Advanced Revelation, please see the chapter "About databases in Advanced Revelation".

Row key lists

Often you will want to access not just one row, but a series of them. To do this, you can use a list of row keys.

For example, you may wish to send a mailing to all of your customers who reside in the state of New York. Rather than having to provide each individual row key to Advanced Revelation, you can have Advanced Revelation build and use a list of row keys corresponding to the rows of the appropriate customers.

Once you have a row key list, Advanced Revelation can pick out each individual row key from the list, and then process the row represented by that key. Common uses for lists of row keys include:

- **Reports.** You might create a list of row keys that represents a subset of the data in a table. This list can then be used when you produce a report with R/LIST, or with the Form or Label processes.
- **Application windows.** You may wish to view or update a selected group of rows in an application window. To do so, you can create a row key list, and then use the list to call each row into the window.
- **Collector windows.** You might use a list of row keys when processing information in a collector window. For example, you might want to copy selected rows from one table to another. In that case, you would use a list of the appropriate row keys in the Copyrow window.

Creating lists of row keys

There are three ways to create lists of row keys:

- Enter them directly. For example, you might do this at the Edit or Copyrow windows.

- Search a table or index using a tool such as Query, EasyWriter, or the SELECT command. When Advanced Revelation "selects" (searches) a table or index, the result, if successful, is a list of keys to the rows meeting the search criteria.
- When entering data in a window. As a user edits rows using a window, Advanced Revelation keeps track of the row keys for those rows. This list is available at any time by using a special keystroke [Alt-U] in the application window.

Types of row key lists

Once a row key list has been created, Advanced Revelation makes it available to you in one of two ways.

Select lists

If you use SELECT or EasyWriter, Advanced Revelation makes the list available as a select list. When a select list is available, it is said to be "active," and the word "SEL" appears in the status line.

In most cases, the next process you execute in Advanced Revelation will use only the rows represented by the currently active select list. As the report or command uses a select list, each key is used and then discarded. When no keys remain on the select list, the report or process is complete.

An active row key list is available for the next process only. If the list isn't used by that process, it is cleared. If you wish to use the active list for several processes, or if you wish to use it in a later process, it can be saved.

Browse lists

In an application window, the row key list operates as a browse list. Typically, the browse list is used to scan ("browse") through a series of rows in order to locate or view the exact row you want. Unlike a select list, the keys on a browse list are not used up. Instead, the browse list operates in a circular fashion. You can page through the rows named on the list, moving forward or back. When you reach the end of the browse list, you circle back to the beginning and continue through again.

Unlike a select list, when a browse list is created you have an opportunity to edit it before using it.

Defining selection criteria for row key lists

When you want Advanced Revelation to create a row key list, you must specify criteria to define which rows in the table will qualify for your list. The selection criteria are specified in a SELECT command.

Entering SELECT commands

You can enter and execute an R/LIST SELECT command in several ways:

- At the Command window
- Within data entry windows, you can use Advanced Revelation's Window Query facility or you can use the Filter key [Ctrl-F10]. (For more information about Window Query, see "Using Window Query" later in this chapter).

The Filter key presents you with several ways to create or activate a selected list of row keys, such as using the Select window or searching an index.

EasyWriter is available in the Select window to help you compose a Select command. While in the Select window, press [F2] or click <Options> to call EasyWriter. For instructions on how to use EasyWriter, see the "Using EasyWriter" chapter.

Details about the options available from the Filter key [Ctrl-F10] follow this section. For more information about setting up selection criteria with a select command, see the SELECT entry in the "TCL Command Reference" chapter in the TCL section.

Using browse lists in application windows

If you are in an application window you can generate an active list of row keys by pressing [Ctrl-F10] and using the Select or Query Table options. You can also have Advanced Revelation provide you with a list of row keys for changed rows by pressing [Alt-U].

Overview of functions

[Alt-U] creates a browse list out of all rows that have been entered or changed during the current session. This permits you to examine changed rows.

The list of keys is initially put into a Browse Edit window. Using the Browse Edit window, you can add and delete keys as desired. When you finish review or editing, press [F9] or click <Save> to make the browse list active. For details about the Browse Edit window, see below.

When a browse list is active, a notice something like this appears in the status line:

```
<0001/0010>
```

Press [Alt-F] to move forward to the next row. Press [Alt-B] to move back to the previous row. Browsing through rows on a browse list is circular. When you reach the end of the list, Browse cycles you through to the beginning again. Similarly, paging back through the first row on the list will place you at the end of the row list.

You can accomplish other functions while using an active browse list. If you wish to edit the browse list, you can press [Alt-I], and the Browse Edit window will appear. You can edit keys in this list normally.

You can also use [Alt-P] to print all the rows in the browse list. If you press [Alt-P], Advanced Revelation will use the current window as a form template, and print each row on the browse list directly to your printer.

Summary of Browse List Keystrokes

Key	Function
[Alt-I]	Display the Browse Edit window
[Alt-F]	Browse forward one row
[Alt-B]	Browse backward one row
[Alt-P]	Print the current list to the printer, using the current form as a template
[Alt-U]	Create a browse list of rows changed during the current session

The Browse Edit window

In an application window, you can display the Browse Edit window at any time by pressing [Alt-I]. The row keys on the active browse list are made available for editing. If no list is active, you can create one by entering row key names. Press [F9] or click <Save> to make the list active.

Browse Edit softkeys

Because the Browse Edit window displays only row keys, it has two softkeys to make it easier for you to edit the browse list. Use the softkeys key [F6] to access each of these softkey options, or execute the softkey directly by pressing the appropriate keystroke.

Choosing and ordering browse keys

[Shift-F1] brings up a multiple-choice popup that makes it easier for you to choose specific keys to include on the popup, and to arrange the keys into a specific order. This is useful when you wish to see only a few keys out of many, and when you wish to see the rows in an order different from the way they appear in the Browse Edit window.

To access this function, press [Shift-F1] at the Browse Edit window. You will see a multiple choice popup with the first page of row keys from the browse list. When using this popup, you should choose row keys in the order you wish them to appear in the browse list.

Viewing additional data for browse keys

[Shift-F2] makes it possible for you to see additional information about each row on the browse list. This is handy when you find it difficult to distinguish the rows on the browse key list by their row keys alone.

For example, your browse list may contain row keys for rows in the CUSTOMERS table. Since you might not know your customers by customer number, you may decide to display the company name and city columns from each row to help you in distinguishing them.

To display the Display Columns Select popup, press [Shift-F2] at the Browse Edit window.

When the Display Columns Select popup appears, it gives you a choice of all columns defined for that table. This is a multiple-choice popup—highlight and select each column to display in turn. When you are through selecting columns, press [F9] or click <Save>.

When you save the columns to be used for browse list display, you will see an expanded version of the Browse List Select popup. This is a list of all keys in the browse list, but it also displays the data in the columns you just chose.

Passing select lists to a window

If a select list is active when a window is invoked, the select list will be passed to the window as a browse list. This makes it possible to pre-select data (for example, using a select or GETLIST command), and then feed the data into a window for review.

This capability is typically used when invoking a window from the Command window or from within an R/BASIC program. For example, if you wish to examine all rows for customers in the state of Kansas, you could type the following two commands in succession at the Command window:

```
SELECT CUSTOMER WITH ST = "KS"
WINDOW CUSTOMER_ENTRY
```

The first command (SELECT) creates a list of all rows in the CUSTOMER table in which the ST column or index contains the abbreviation KS. Once these rows have been found, the window CUSTOMER_ENTRY is invoked. The rows found by the SELECT command will become a browse list in the window.

Note Although a select list (filter) can be any length, a browse list is limited to a length of approximately 32K (32,768 bytes). If you create a very large select list and pass it to a window, the window will truncate the select list to approximately 32K before turning it into a browse list.

Using the Filter key

To create and manage row key lists within any collector or application window, press the Filter key [Ctrl-F10]. This produces a popup of selection options.

Depending on where you are in Advanced Revelation when you press the Filter key, you will see slightly different options for creating row key lists. If you are in a collector window or application window for an unindexed table, you will see the Browse Selections popup. If you are in an application window for an indexed table, you will see the Browse Options popup. Both popups are similar, but the Browse Options popup has additional capabilities.

Browse Selection options—popup

When you press [Ctrl-F10] in a collector window or in an application window that accesses an unindexed table, Advanced Revelation displays the Browse Selections popup. The various popup options present you with different methods of creating lists of row keys.

If you are in a collector window, any key list created or activated using this popup operates as the active select list for the current window. If you are in an application window, the selected list of row keys will function as a browse list. For details on using a browse list, see "Using Browse Lists in Application Windows" above.

These options appear on the Filter key popup for Browse Selections:

- Select
- Query Table
- GetList
- SaveList
- Filter

Each option on the Selections popup is described below.

Select

Choose the Select option to access the Select window. The window assists you in creating a Select command. Once you have finished specifying your selection criteria using this window, press [F9[or click <Save> to search the table.

For information on how to specify selection criteria using the Select window, see the "Using EasyWriter" chapter.

Query table

Choose the Query Table option to access the Stored Query Table. This table contains the most recent Select commands executed by Advanced Revelation. It provides a convenient way to retrieve the list of row keys generated by any of the Select commands in the table. Choose the desired Select command and press [F9] or click <Save>.

You can control how many lists are stored in the Query Table. Choose **Options-Configuration-Environment-General** from the Opening menu, and at the *Number of saved queries* prompt, enter a new value.

Getlist

Choose the GetList option to access the GetList window. Use it to retrieve a list of row keys stored earlier with a Savelist command. Enter the name of the list you want to use and press [F9] or click <Save>.

Savelist

The Savelist option allows you to save a row key list so that it can be used at a later time. When you choose this option, the Savelist window appears. Give the currently active list an identifying name and press [F9] or click <Save>. The list is stored as a row on the LISTS table.

A saved list can be used over and over by reactivating it with the Getlist command. This is useful if you wish to run several reports or processes using the same list of row keys.

Note The command or process that uses a select list always uses up the list. Therefore, if you intend to save the list, you must use the Savelist option before applying the select list to any process.

Filter

The filter option is used to run a filter (select statement) that you stored earlier using Query mode in a window or using EasyWriter.

Filter key options—Browse Selections

When the Filter key is pressed in an application window for an indexed table, Advanced Revelation produces a popup labeled Browse Options. Any key list created or activated using this popup operates as a browse list within the application window.

The Browse Options popup enables you to execute the same options as the Browse Selection popup discussed above, but gives you the additional capability of creating row lists out of indexed columns.

These options appear on the Browse Options popup:

- Indexed Columns
- Browse Selections

Use the first option in the popup to see which columns in the table are indexed and to select rows based on an indexed column.

Use the second option to create a row key list from unindexed columns. Option 2 calls the Browse Selections popup menu, which is identical to the Browse Selections popup described above.

The Indexed Columns option

If you choose the Indexed Columns option from the Browse Options popup, you will be presented with the Index Selection window, which contains prompts for each indexed column in the current table. If a column has been indexed using a Cross Reference index, the word "XREF" will be appended to the column name to alert you to this fact.

To create a row key list out of an indexed column, move the cursor to the column on which you wish to base your selection, and fill in the value you want to search for in the column. You can specify more than one value, and Advanced Revelation will find the rows containing any of the values you enter.

Note You can search indexes not just for exact matches of data, but for substrings, ranges, and specific patterns. These index search capabilities, and the index lookup syntax are described later in this chapter under "Using Indexes".

For example, assume you were working in a window containing a column called "STATE", which had been indexed. You wish to find all rows in which the state column contained either "CA" or "AZ". You would call the Browse Options popup using the Filter key [Ctrl-F10], then choose the Index Selection option, which would present you with the Index Selection window.

Move the cursor to the STATE prompt of the Index selection window, and type in the word "CA" and then the word "AZ". When you save your choice, Advanced Revelation searches the index and creates a browse list out of all the row keys associated with rows containing either "CA" or "AZ" in the state column.

If Advanced Revelation finds just one row matching your lookup criteria, it fills the row into the window. However, if there is more than one row corresponding to your lookup criteria, it will present you with the Browse Edit window. Use of browse lists and the Browse Edit window is explained earlier under "Selecting and browsing through data".

The Browse Selections option

The Browse Selections option of the Browse Options popup calls up a popup of methods by which you can create row key lists. This Browse Selections popup is identical to the Browse Selections popup. Refer to the discussion above for details on how to use this popup.

Using Indexes

If you are not familiar with Advanced Revelation Btree and Cross Reference indexes, please see the chapter "Creating and maintaining indexes".

About index search options in application windows

If you have established indexes for your data tables, you can make use of these indexes when looking up data in windows.

- You can use \ (backslash) plus a search value directly at the key prompt. This creates a direct index search.
- You can search indexes not just for exact matches of data, but for substrings, ranges, and specific patterns. The Browse Select window, accessible from the Filter key [Ctrl-F10], prompts you for information used to search the indexes.

Rows containing the information that you want are listed on browse lists. Once a browse list is active, it can be used in the window to view the rows. For more information on using browse lists, see "Selecting and browsing through data" earlier in this chapter.

Index lookup at key prompts (\)

You can initiate an index lookup in an application window directly from the key prompt. This saves you the trouble of first pressing [Ctrl-F10] (Filter), choosing the Indexed Columns option from the resulting popup, and filling in data at the Browse Cross Reference window.

To initiate a direct index lookup in an application window, enter \ (backslash) plus the data to look up at a key prompt. For instance, to create an index lookup based on the word ACME, you would enter the following at the key prompt in a window (assume that Customer No is the key prompt):

```
Customer No \ACME
```

If more than one column is indexed in the table, you will see a popup of all indexed columns. Choose the column for which you are searching. For example, if you wish to look up the value ACME in the COMPANY_NAME column, choose this column

name from the popup. If only one column is indexed in the table, no popup of column names appears.

Any row keys found by the index lookup are returned to the application window as a browse list.

Index lookup syntax

You can create browse lists in application windows using comparison, substring, range and pattern match searches. This means that you can look up data from the indexes if you know only part of a word that was indexed. You can also create a search based on a range of values, or on a pattern that describes the data.

These search capabilities are available anytime you use the Browse Cross Reference or Shared Index Information windows to search an index. This type of index search technique is often used in application windows. Successful index lookups return a browse list to the application window.

There are three areas in Advanced Revelation that are affected by this index search capability:

[Ctrl-F10]-Indexed Columns

This Filter key and popup combination, called from an application window, displays the Browse Cross Reference window, enabling you to enter index search criteria. For details on using [Ctrl-F10] with indexes to create a browse list, see "Filter key options—popup" earlier in this chapter.

\ (Backslash)

A backslash followed by data at the key prompt of an application window works much like the [Ctrl-F10] Indexed Columns option. However, the backslash initiates an index lookup directly, bypassing the Browse Cross Reference window. This capability is described under "Index lookup at key prompts" earlier in this section.

Code and Command

The "B" code and command sequence displays a Shared Index Information window under developer-specified circumstances (for example, using the Options key). This window prompts you to enter index search criteria. For more information on the "B" code and command, see the "Codes and Commands" chapter.

Using index lookup syntax

When accessing indexes and creating browse lists in one of these ways, you can use special characters to create substring, range, or pattern match searches. The index search methods available are listed in the following table:

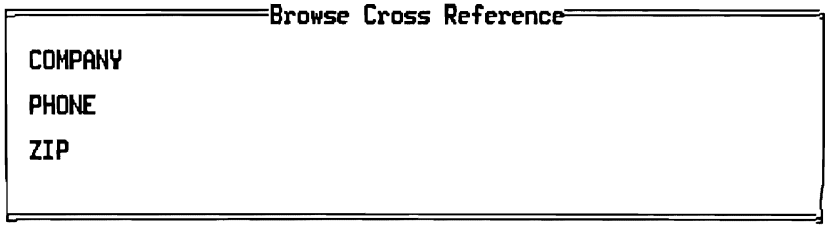
Index Search Methods

Operation	Method	Operator	Example
Comparison	Equal	=	=50
	Greater than	>	>99
	Less than	<	<100
	Greater than or equal to	>=	>=100
	Less than or equal	<=	<=1000
Substring	Beginning with	...]	ACM]
	Ending with	[...	[ING
	Containing	[...]	[TON]
Range	From-To	...	25...50
Pattern Match	Matching	%	%3N'–'4N
Logical	Not	#	#>1000 (not greater than 1000)

Search examples

Imagine that you have indexed the columns COMPANY, ZIP, and PHONE in the CUSTOMER table. You are at an application window that you use to enter and view customer data.

When you press [Ctrl-F10] to access the Browse Selection popup, your first option will be Indexed Columns. If you choose this option, a window will appear with the column names in it, similar to this:



If you wish to look up all companies in which a portion of the company name is ACME, you would fill in the COMPANY prompt like this:

```
COMPANY   [ACME]
```

This creates a search for all companies in which the string ACME appears anywhere in the name. If you were only interested in companies whose name began specifically with the word NORTH, you would fill in the prompt like this:

```
COMPANY   NORTH]
```

In another example, if you wish to look at all customers within a certain ZIP code range, you might fill in the window like this:

```
ZIP    90000...99999
```

This creates a search for all customers whose ZIP code falls in the range 90000 to 99999, inclusive.

Finally, you might wish to find all customers outside your local telephone area. You can identify these customers easily, since you entered no area code for local customers. You might fill in the window like this:

```
PHONE %3N'-'3N'-'4N
```

This creates a search for all customers whose telephone number consists of three numbers, a dash, three numbers, another dash, and four numbers. In other words, you are looking for customers whose phone number matches the pattern of area code plus number.

Note For more information on creating match searches, see "Data validation" in the chapter "Creating windows with Paint".

Searching with multiple criteria (AND and OR)

If you create a search using multiple criteria, Advanced Revelation will make certain assumptions about how to construct your query.

If you specify multiple search values for a single column, the query will assume a logical OR between these values. In other words, the query will search for rows that contain *any* of the search values. For example, you might enter search values such as these:

```
COMPANY   ACME
          HAUSER
```

This creates a search for all rows in which the company name is either ACME or HAUSER.

On the other hand, if you enter search values into more than one column, the query will assume a logical AND between columns. This means that you will be requesting

rows that meet *all* search criteria. For example, you might create a search using these values:

```
COMPANY   ACME
   ZIP    >=50000
```

In this case, the query will be constructed to search for those rows in which the company name is ACME and the ZIP code is greater than or equal to 50000. Any rows to be found must meet both criteria.

Reserved characters

Occasionally, you might wish to base a search on data that includes the special characters listed in the table "Index Search Methods" earlier in this section. For example, you might wish to find a company whose name includes the "%" (percent character).

If you look at the table, you will find that % (percent character) is a reserved character meaning "search for a specific pattern". This causes a problem, since the Browse Cross Reference window will interpret your percent character incorrectly.

In order to overcome this problem, you can repeat a reserved character to indicate that it is a part of the search data. If the reserved characters used in the Browse Cross Reference window are repeated, they are treated as literal characters, not special characters.

For example, imagine that you have a customer whose company name is "The %-age Company". If you try to create a search using the syntax:

```
COMPANY   %-age
```

the Browse Cross Reference window will interpret your search as a matching search, and will attempt to resolve "-age" as a match criterion. However, you can use this syntax:

```
COMPANY   %%-age
```

Here, the two percent characters together indicate that the percent character is to be treated as a literal character, and not as a matching search.

Repeating characters in this fashion works for all the reserved characters except the ellipsis (...). This symbol cannot be made a part of the search data.

Using Window Query

With Advanced Revelation's Window Query facility, you can easily look up rows from a data table in the same windows you use for data entry. With Window Query, any

data entry window becomes a form that you simply fill out with the search and sort criteria you wish to use.

Window Query takes the information you enter into a data entry window and constructs a Select command. When you press [F9] or click <Save>, the Select command is run. Any rows found by Window Query are returned as a browse list into the data entry window.

Window Query supports most R/LIST selection features. You can specify selection criteria, including all operators such as equal, less than, greater than, starting with, ending with, containing and matching. Window Query also supports multiple selection criteria linked with AND and OR, as well as multiple sort criteria.

Starting Window Query (\)

To use Window Query in any data entry window, enter \ (backslash) then press [Enter] at the key prompt. This will turn the window into a form in which you can move the cursor from prompt to prompt, entering selection and sort criteria.

In Window Query, the title of the screen in the top border adds the words "– Query Window" after the original title of the window. This indicates that you are in Window Query.

Moving between prompts

While in Window Query, you can use the cursor arrow keys ([↑], [↓], etc.), the [Enter] key, or [Alt-A] to move from prompt to prompt. These are the same keys that are active in a normal data entry window.

If your window can normally use prompt or page tabs, you can use these in Window Query as well. For example, if you can use [Alt-Tab] and [Alt-↑] (backtab) in your window, these keys will also be active in Window Query.

You can move into symbolic prompts as well as normal prompts, but you cannot search on prompts that are defined in the window as joined columns.

For more information on cursor movement in windows, see the chapter "Navigating Advanced Revelation".

Creating the Window Query

To create a query, initiate Window Query by pressing \ (backslash) at the key prompt. Then move the cursor to the appropriate prompt(s) and enter the data to be used for the query.

For most prompts, Window Query will accept input at the prompt entry line. When the cursor moves to that prompt, Window Query will display a special character mask that is as long as the prompt entry line defined in the window.

Hidden prompts or prompts for which the label has been deleted will prompt for input in the bottom border of the window. In place of a prompt label, Window Query will use the dictionary name associated with that prompt.

If you need more space to enter selection and sort criteria, use the [F3] key to zoom the prompt into a separate window. This gives you more room to enter information. After you have entered selection and sort criteria, press [F9] or click <Save> to save your changes and exit the Zoom window. You will be returned to the Window Query window.

Press [Esc] at any prompt to quit Window Query and return to the original data entry window.

Running the Window Query

Once all selection and sort criteria have been entered, press the [F9] key or click <Save>. Window Query will construct and execute a Select command.

Any rows found by the query will be returned as a browse list into the original data entry window. If no rows are returned, Window Query returns to the original data entry window with no browse list. Any browse list you might have had active before you began your query will be cleared.

Entering selection criteria

For most searches, simply enter the data at the prompt. To specify an operator other than = (equal), enter the operators in front of the search data.

For substring searches (starting with, ending with, or containing), or for range searches (from..to), embed the operators for these searches directly into the search data.

The full set of operators available in Window Query is listed below.

Operators Available in Window Query

Operation	Method	Operator	Example
Comparison	Equal	=	=50
	Greater than	>	>99
	Less than	<	<100
	Greater than or equal to	>=	>=100
	Less than or equal	<=	<=1000
Substring	Beginning with	...]	ACM]
	Ending with	[...	[ING
	Containing	[...]	[TON]
Range	From-To	...	25...50
Pattern Match	Matching	%	%3N'–'4N
Logical	Not	#	#>1000 (not greater than 1000)

For example, to enter criteria at a prompt called Company, you might enter one of the following:

Data	Meaning
ACME	Company equal to ACME
ACM]	Company starting with ACM
>=ACME	Company greater than or equal to ACME
%5A	Company matching 5 alpha characters
#[INC]	Company not containing INC
ACME...DIAMOND	Company from ACME to DIAMOND

For more information on the use of operators (comparison words) in selection criteria, see WITH in the "R/LIST Keyword Reference" chapter in the R/LIST section.

You should enter information at the prompt exactly as it appears in the database. For instance, if you have stored names in your database using upper and lowercase (for example, Smith), your search should be constructed the same way. Conversely, if data in the table is all uppercase (for example, SMITH), the information in the Window Query window should be entered in uppercase.

The data need not be in quotes; if quotes are embedded in a search value, they will be included as a part of the search. Beginning and ending spaces are trimmed from the data, but embedded spaces are considered a part of the search data.

Multiple criteria for a single prompt

You can enter more than one selection value for a single prompt by separating each value with ; (semicolon) or & (ampersand).

Separating search values with ; (semicolon) will cause the values to be linked with a logical OR in the search. Separating the values at a prompt with & (ampersand) will cause those values to be combined with a logical AND.

For example, you might construct searches like the following:

```
Company   ACME ; MILLER
Contact   [SMITH] & [JACK]
```

The first example will search the table for all rows in which the company name is ACME or MILLER; either would be acceptable.

The second example will search the table for those rows in which the contact name contains both the name SMITH and the name JACK. In other words, the search will find rows in which the contact name is JACK SMITH.

The second example also illustrates that you can use operators (for example, [ ], <, etc.) when linking data. Simply include the operator by placing it directly in front of, or embedded in, the data. For example:

```
ZIP    <10000 ; 80000...90000
```

Using AND and OR between prompts

If you enter selection criteria for more than one prompt, Window Query links these prompts with AND. For example:

```
Company   ACME
   ZIP    50000
```

Window Query creates a search in which the company name is ACME and in which the ZIP code is 50000. Both criteria must be met for a row to be selected.

If you wish to link selection criteria for different prompts with an OR, use a semicolon (;) as the first character at that prompt. For example:

```
Company   ACME
   ZIP    ;>50000
```

This query will find rows in which the company name is ACME or the ZIP code is greater than 50000. If either criterion is met, the row will be selected.

Entering sort criteria

In addition to creating selection criteria, Window Query can include specifications for sorting. To indicate that a prompt is to be used in sorting, use the keyword BY. The keyword BY can be included anywhere in the data entered for a prompt. For example:

```
ZIP    BY
```

Entering the word BY in this way will cause Window Query to create a search in which the rows will be sorted according to ZIP code.

If you wish to sort by multiple criteria, all BY keywords should be followed immediately by a number for the sort priority. Thus, to indicate that a prompt is to be the primary sort prompt, the phrase should read BY1. Indicate additional sort levels by including the phrases BY2, BY3, etc., at different prompts. For example:

```
ZIP       BY1
Status    BY2
```

In this query, the rows will be sorted first by ZIP code, and then within like ZIP code values, by STATUS.

To create a sort in descending (reverse) order, use the keyword BY-DSND instead of BY. For example:

```
Order Date  BY-DSND
```

This query produces a report in which the rows appear in descending order — in other words, with the most recent order date first.

If more than one descending sort is required, you can indicate a sort priority. The convention is the same as for BY. For instance, to indicate that the second-level sort is to be descending, enter the keyword BY-DSND2 at a prompt.

Editing the Select command [Shift-F1]

Press [Shift-F1] while in Window Query to display the Window Query Select window. This window contains the Select command that Window Query has constructed using the search and sort criteria you entered.

You can use the Window Query Select window to edit the command. For example, you might wish to change the sort order of prompts. Rather than entering and changing the appropriate prompts in the Window Query window, you could do this in the Window Query Select window. You might also wish to change the parentheses that Window Query builds in the Select command.

If you are an R/LIST novice, you can use Window Query as an R/LIST tutorial. Construct searches using Window Query, and call the Window Query Select window often to display the correct syntax of the resulting R/LIST command.

To save any changes you have made in the Window Query Select window, as well as to execute the Window Query, press [F9] or click <Save>.

Note If you change the Select command while in the Window Query Select window, you cannot return to Window Query in the data entry window.

When you save the command, you will be given an opportunity to save your query as a filter. See "Saving the Window Query" below for details about this function.

If you press [Esc] at the Window Query Select window, you will be returned to the Window Query entry window, and can continue to enter selection and sort criteria. If you have made changes while in the Window Query Select window, these changes will not be saved.

Saving the Window Query [Shift-F2]

At any point while you are creating a search in Window Query you can save the current query. To do so, press [Shift-F2]. A window will appear prompting you for the name of the filter under which you wish to save the query.

Enter a name without any spaces. The Select command that Window Query has created out of your search criteria will be saved using the name you specify.

Redisplaying the last Window Query

Window Query remembers the last query you constructed and executed. You can recall this last query if it was created in the same window and if you have not logged off or changed applications since you created it.

To recall the entire last query (except for changes you made in the Window Query Select window), enter Window Query by typing in \\ (two backslashes) and [Enter] at the key prompt. Window Query will display all the selection and sort criteria you last used.

You can also recall the last query by entering Window Query normally using \ (single backslash). Once you are in Window Query, press [Alt-C] to recall the entire last query.

To recall the selection or sort criteria for a single prompt, press [Alt-O] at that prompt. Window Query will fill in the criteria you last used for that prompt.

Changes to prompts in Window Query

Because you are not truly entering rows when in Window Query, some prompt specifications and window processes for the window are not active. The following

changes are made to prompts and other processes in a window while in Window Query:

- Multivalued and text prompts are treated as single-valued, one-line prompts.
- Output patterns and output masks are ignored.
- Pattern matches are disabled.
- Maximum length specifications are ignored.
- All prompts accept lowercase input.
- Individual detail help messages or processes are replaced with global Window Query help.
- Window-specific softkeys are replaced with softkeys for the Window Query process.
- The Relations key is disabled.
- Pre-prompt and post-prompt processes are disabled.
- Pre-processes, post-processes, and replace-processes are disabled.

Accessing indexes in Window Query

If you have created indexes for your data table, you might be able to take advantage of them in your query. If so, the query will run faster.

Accessing Btree indexes

Window Query will automatically make use of Btree indexes whenever practical. When you are searching or sorting a column, Advanced Revelation will determine if the column is indexed using a Btree index, and if so, will determine if it is efficient to use the index. You do not need to make any special provisions or add any commands in order for Advanced Revelation to use a Btree index.

For example, if you create a search in Window Query that looks for a particular range of ZIP codes, and if the ZIP code column has been indexed using a Btree index, the query will determine if it is efficient to use the index to find the appropriate rows. If so, the query will simply find the rows in the index. If ZIP code is not indexed in a Btree index, the query will be forced to look at every row in the table, examining each one in turn to determine if it fits the search criteria.

Accessing Cross Reference indexes

In contrast to Btree indexes, Cross Reference indexes are not automatically used when you create a query. However, if you know that a particular column has been indexed using Cross Reference indexing, you can specify that the query use the index.

To specify that you wish the query to use a Cross Reference index, put a \ (backslash) in front of the search data.

For example, entering this at the company name prompt:

```
Company   \ACME
```

will cause the query to look for the word ACME in the Cross Reference index. Under most circumstances, this will be considerably faster than looking through each row in the data table.

If you use a \ (backslash) in front of search data, Window Query will check whether the column has been indexed using Cross Reference indexing. If it has not, Window Query will simply ignore your request to use the index. Since this is the case, you can use a backslash anytime you suspect that you might be able to take advantage of Cross Reference indexing, even if you are not sure.

If you want to search for multiple values in a column, and you want to use a Cross Reference index for each of the values, each value in your entry must be preceded with \ (backslash). For example, if the column COMPANY is indexed using a Cross Reference index, and you want to search for the company names ACME and ABC, you would enter \ before each of these values:

```
Company   \ACME; \ABC
```

If you want to include an operator (<, >, >=, etc.) when doing a search in a Cross Reference index, the operator should follow the \ (backslash). For example, this entry at the Company name prompt (COMPANY is assumed to be indexed using a Cross Reference index) will create a search for rows in which each word in the company name column (less those in the stop list) starts with a letter higher than M:

```
Company   \>=M
```

Note To determine whether data has been indexed in a Cross Reference index, produce a list of indexes by choosing **DB Admin-Indexes-List** from the Development menu.

Cross Reference indexes and stop lists

When you use a Cross Reference index in a query, you should be aware of the stop list used in that index. A stop list is a list of words that are considered too common to index when the Cross Reference index is created. A stop list typically includes words such as "THE," "INC," "AND," etc.

If you inadvertently use a Cross Reference index for a query involving a word on a stop list, the query will not find the word, since it will not be in the index.

For more information on stop lists, see the chapter "Creating and maintaining indexes".

Using Window Query at the Command window

You can call a window directly from the Command window and display it with Window Query active. To do so, enter the command QUERY followed by the name of the window. For example, the command:

```
QUERY CUSTOMER_ENTRY
```

will cause the CUSTOMER_ENTRY window to be displayed in Window Query. You will not need to enter a \ (backslash) in order to put the window into Window Query.

Enter selection and sort criteria normally. When you have finished constructing your query, press [F9] or click <Save> to save the query and execute it. If you press [Esc] before executing the query, you will be returned to the Command window.

When the query is finished, you will be returned to the Command window. The select list generated by the query will be returned to the Command window as an active select list, rather than being returned as a browse list into the data entry window. The active select list can then be used in a subsequent process, such as a LIST command, an R/BASIC program, etc.

If your query produced no results, there will be no select list active when you return to the Command window.

Examples of Window Query Syntax

Prompt	Data	Meaning
State	#WA	State not equal to WA
Company	[COMPUTER]	Company containing COMPUTER
ZIP	#>70000	ZIP not greater than 70000
Phone	%3N'–'3N'–'4N	Phone matching the pattern: 3 numbers, dash, 3 numbers, dash, 4 numbers
Invoice Date	1/1/87...12/31/87 BY	Invoice date from 1/1/87 to 12/31/87 by invoice date
Contact	\JAMES	Contact equals JAMES (force Cross Reference index lookup, if the index exists)
Status Last Order Date	INACTIVE ;<1/1/86	Status = INACTIVE or last order date less than 1/1/86
City State Area Code	A] ; B] BY2 BY1 BY3 <300	City starting with A or City starting with B and Area code less than 300, by State by City by Area Code
President	[TRUMAN] ; [QUINCY] & [ADAMS]	President containing TRUMAN or President containing QUINCY and ADAMS

Quick Reference Chart of Window Query Syntax

Operation Type	Operator	Operation
Entering and Exiting Window Query	\	Enter Window Query
	\\	Recall last query
	[Alt-C]	Recall last query
	[F9]	Execute query
	[Esc]	Exit without saving
Moving between prompts	[Alt-A]	Move to selected prompt
	[PgDn] / [PgUp]	Move to next/previous prompt
	[Enter]	Move to next prompt
Editing in Prompts	[F3]	Expand prompt edit area (zoom)
	[Alt-O]	Copy prompt value from previous query
Selection Operators	= or null	Equal
	>	Greater than
	<	Less than
	=	Greater than or equal to
	<=	Less than or equal to
	#	Not (not equal to)
	]	Starting with
	[	Ending with
	[ ]	Containing
	...	From ... to
	%	Matching
	;	OR
	&	AND
	\ data	Use Cross Reference index for searching, if applicable
Sort Criteria	BY	Sort BY this column
	BY*n*	Sort BY this column, priority *n* (example,BY2 = secondary sort column)
	BY-DSND	Sort BY this column in descending order
Softkeys	[Shift-F1]	Display Window Query Select window (edit search command)
	[Shift-F2]	Save Window Query as filter

15. Using the table browser

About the table browser

Use the table browser to view data in a browse list. Each row in the table browser displays a row of data in a table.

Accessing the table browser

Accessing the table browser from a data window

1. Activate a browse list. You can create a browse list within a data window using window query or enter the SELECT command at the Command window. SEL will appear in the status line.
2. Press [Ctrl-F5].

Accessing the table browser from the Command window

At the Command window, enter:

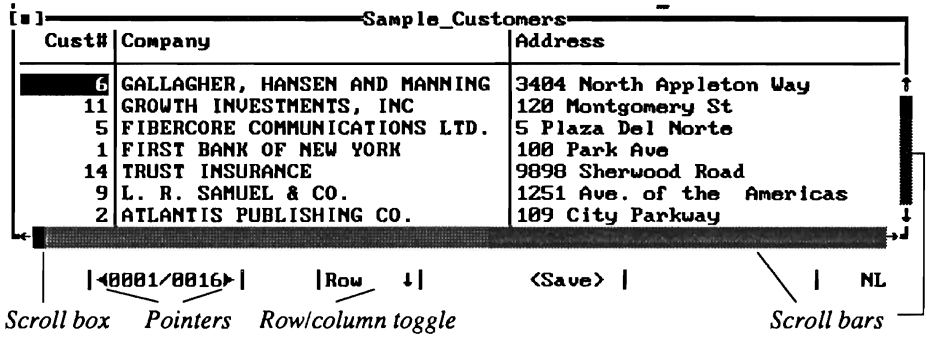
```
BROWSETABLE table column(s) BY clause(s) WITH clause(s)
```

Table is the name of the table you want to browse and *columns* are the names of the columns you want to include in the table browser. Use the BY clause to order the data and the WITH clause to limit the rows you are displaying.

For more information, see SELECT in "TCL Command reference" in the *Reference* manual.

Navigating in the table browser

The table created by the table browser may be larger than the visual display area. If the table is larger than the screen, there will be scroll bars on the right border, the bottom border, or both borders.



Using the scroll bars

Drag the scroll box to a position in the scroll bar corresponding to the row or column where you want to view data.

Active keys in the table browser

If you want to	Press
Move to the first cell in a row	[Ctrl-Home]
Move to the last cell in a row	[Ctrl-End]
Move to the first cell in the first row	[Ctrl-PgUp]
Move to the first cell in the last row	[Ctrl-PgDn]
Move to the first cell in the row displayed	[Home]
Move to the top/bottom of a page	[PgUp] \ [PgDn]
Move to the last cell in the row displayed	[End]
Move right/left one cell in row	[←] \ [→]
Move up/down one cell in column	[↑] \ [↓]

Moving within a row or column

When you press [Enter] in a cell, you can move to the next cell in the row or the next cell in the column. The status line indicates whether you are moving within a row or column. To change how the cursor moves:

Click the Row/column toggle.

1. Press [F2] to display the Table Browser Options popup.
2. Highlight the Row/Column option.
3. Press [Enter], then [F9] to save your selection.

Changing the display of the table browser

Use Paint to change the location and size of the table browser when you press [Ctrl-F5] in an application window. You can also change display characteristics for a specific column when it displays in the table browser.

Controlling table browser display in a window

To modify how the table browser displays from an application window, choose the **Advanced-Table Browser** option from the Paint menu. In the Table Browser window, you can specify:

- The size and location of the table browser at the *Left Column, Right Column, Top Row* and *Bottom Row* prompts. The table browser can be up to 80 columns wide and 23 rows high.
- One of five border types at the *Border Type* prompt.
- The title of the table browser at the *Border Title* prompt.

Press [F9] or click <Save> to save your entries in this window.

Changing column display for [Ctrl-F5]

By default, if you press [Ctrl-F5] in an application window, every prompt in the window will display as a column in the table browser. The default column heading will be the prompt's label. The column's width will be the display width of the prompt.

However, you can decide whether a column that has a related prompt in an application window will display in the table browser. You can also determine the width of the column in the table browser and its column heading.

To remove a column from the table browser:

1. From the Development menu, choose the **Define-Window** option.
2. Display the window you want to modify.
3. Highlight the prompt you wish to remove from the table browser.
4. Press [Shift-F6] to display the Advanced Prompt Options window.
5. Move your cursor to the *Browser Disable* prompt and enter "Y".
6. Press [F9] or click <Save> to save your changes.

To change column characteristics in the Table Browser:

1. Complete steps 1 through 4 above.
2. Move to the *Browser Length* prompt and enter the number of characters to display.
3. Move to the *Browser Heading* prompt and enter a column heading. The column heading does not have to be enclosed in quotes.
4. Press [F9] or click <Save> to save your changes.

Controlling the default column display characteristics in table browser

If a column is included in the table browser, its default appearance is based on display characteristics established in the column's dictionary entry.

To change the dictionary entry while in Paint:

1. Highlight the prompt and press [Shift-F10].
2. In the Dictionary window, move to the *Column heading* prompt and enter the default column heading.
3. Move to the *Display length* prompt and enter the default display length.

4. Press [F9] r click <Save> to save your changes
5. Press [Esc] to return to the window.

For more information, see "Managing tables" in the *User's Guide*.

You can also display the dictionary window by choosing the **DB Admin-Columns-Define** option from the Development menu.

Note If you change a column's dictionary entry, these changes will be reflected in all reports that use this dictionary information.

16. Creating Forms

Creating Forms

Advanced Revelation's Paint tool enables you to create Forms to match your data to an existing paper form or to create a layout to your own specifications. While using Paint to create a Form, you can also define the table that will hold the Form's data. This chapter describes how to build Forms using Paint. For more information on Paint, see the chapter "Creating Windows with Paint" in this book.

Form reports are commonly used when you want to place columns in a particular layout. You use Paint to create Forms in much the same way as you use it to create windows. This section describes the features in Paint as they relate specifically to Form definition.

While defining your Form, you determine the display width of each column, the number of rows printed across the Form, and how many spaces Advanced Revelation leaves between rows. In this way, you determine the size of the printed Form.

Using Forms to design table and row structure

If you are creating a new application or table, you can use a Form design to help you design the table that holds its data. In fact, you can design all your reports first, and then design the application based on its reports.

This approach is particularly convenient when building Form reports, because you can create your table at the same time as you create your Form.

First, determine what information will appear on the Form. Decide where and how you want the information displayed. Next, create your Form and table at the same time in Paint.

Creating a Form

To create or modify a Form, choose **Define-Report-Form** from the Development menu. Advanced Revelation asks you whether you will be creating new Form or editing an existing one.

Modifying an existing Form

If you want to modify an existing Form, answer with "Existing". Paint displays the Paint Form window, which asks you to enter the name of the Form that you want to modify. Press [F2] or click <Options> to see a list of available Forms.

Creating a new Form

If you want to create a new Form, respond with "New". Paint then asks you whether you will be creating a new table or using an existing one.

Using an existing data table

To create a Form that will access an existing table, choose the name of the table from the popup.

Creating a new table

If you tell Advanced Revelation that you want to create a new table, Paint displays the Make Dictionary window. Fill in the name of the table that you want to create. At the *Location* prompt, enter the location where you want the table to reside. Press [F9] or click <Save> to save your changes and proceed to Paint.

Using virtual space to plan your page size

You are not limited to the size of the screen in creating your report. In fact, the maximum area you can use is roughly 200 characters wide and 160 characters deep. The size of your printed Form is limited only by the size of paper your printer can handle, and the amount of memory you have available.

Using Paint options to create Forms

From this point, creating Forms with Paint is almost identical to creating windows. Therefore, you should follow the documentation in the chapter "Creating windows with Paint".

There are several Paint options that apply specifically to painting Forms. These are discussed below.

Hiding prompts

You can suppress the display of a prompt on your Form while retaining the prompt's display lines. Use this feature when you want to display the data without any accompanying text or with a label heading.

To hide a prompt, follow these steps:

1. Move the cursor inside the prompt and select it with the mouse or by pressing [Space].
2. Press [F2] or click <Options> to display the Prompt Options popup.
3. Choose **Hide label** from the Prompt Options popup. Paint hides the prompt.

Showing column names in display lines

After hiding prompts on your Form, you may want to view the column names in their display lines. To display the column names, choose **Advanced-Overstrike** from the Paint menu.

When Paint displays your Form, the column names appear in their display lines. You can continue defining your Form with the column names displayed. If you edit a display line, the column name in the display line disappears. To display it again, repeat the above steps.

Changing prompt characteristics

You can determine how Advanced Revelation displays the data by setting display length, justification, and output format.

Changing the display length

Display length refers to the size of the prompt's data display line. Advanced Revelation displays data on the Form within the dimensions of the display line. To change the display length of a prompt, select the prompt entry area, then use [←] and [→] to change the size of the entry area.

Displaying the prompt window

The Prompt window is used to adjust a prompt's *Justification* and *Output format* characteristics. To see the Prompt Details window, select the prompt, then press [Shift-F5].

Press [F1] at any prompt in the Prompt Detail window for more information. Press [F2] or click <Options> for options to enter at the prompt. When your entry is complete, press [F9] or click <Save> to save it.

Adjusting prompt justification

The *Justification* prompt in the Prompt Detail window controls the justification for the prompt's data. These are the format options (the default setting is left justified):

Option	*Description*
L	Left justified. The data starts at the left border of the display line. Data that extends beyond the right border is not displayed.
R	Right justified. The data ends at the right border of the display line. Data that extends beyond the left border is not displayed.
T	Continuous text. Text will be formatted exactly as it was entered.
TN	Text nonjustified. Any special text formatting will not be retained.
C	Centered. The data is centered between the borders of the display line.

Changing a prompt's output format

Use the *Output format* prompt in the Prompt Detail window to format the display of date, time, currency, decimal, and other data with a special format (phone numbers, Social Security numbers, zip/postal codes, and the like).

The *Output format* prompt sets a specific format for the output of the data, regardless of how the data was entered. For example, you can set this pattern for the output of a date: mm-dd-yy. Even if the date was entered in a different format (for instance, dd month yyyy), the Form report displays the date in the pattern you set.

For more information on output formats, see the chapter "Creating windows with Paint".

Specifying Form characteristics

Just as you use prompt characteristics to determine how data for an individual prompt will be displayed on your Form, you use Form characteristics to determine how the entire Form will be printed or displayed on your screen.

To define your Form's characteristics choose the option **Advanced-Form parameters** option on the Paint menu. Paint displays the Make Form Parameters window. The specifications at this window help you organize the information on your Form. When you are finished making entries at the window, press [F9] or click <Save> to save them.

Determining how many rows to print on a page

By default, Advanced Revelation prints one data row on each page of your report. To print more than one data row across your page and down your page, you must specify the number.

Number of rows across page

To print more than one data row across a page, enter the desired number at the *Number of rows across page* prompt. Advanced Revelation prints as many data rows as you determine, side-by-side across the page.

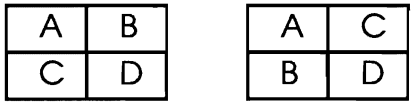
Number of rows down page

To print more than one data row down a page, enter the desired number at the *Number of rows down page* prompt.

Arranging the rows in the report

If your Form displays more than one row across the page, you can tell Advanced Revelation to arrange it by row or by column. Arranging by rows means that the data

rows will be in sorted order across the page; arranging by columns means that the data rows on the report will be in order down the column:



Arranged by rows *Arranged by columns*

By default, Advanced Revelation orders by rows. Use the *Row order* prompt to change your row order.

Arranging by rows across the page saves printing time. Advanced Revelation can begin printing as soon as it has formatted one row. If your rows are ordered down the page, Advanced Revelation must format the entire page before it can begin to print.

Formatting a Form

You can specify a variety of formatting options, including the number of spaces between rows, the top and left margins, and headings and footings.

Columns between rows

If your Form displays more than one data row across a page, use the *Columns between rows* prompt to specify the number of spaces between the data rows. By default, Advanced Revelation puts two spaces between data rows.

Lines between rows

If your Form displays more than one data row down a page, use the *Lines between rows* prompt to specify the number of blank lines between rows of rows. By default, Advanced Revelation puts two blank lines between the data rows.

Top margin

Use the *Top margin* prompt to set your Form's top margin. By default, Advanced Revelation prints the first line of your Form on the fifth line from the top of the page.

For example, if you're printing on letterhead stationary and want to start printing on line 10, enter 10 at the *Top margin* prompt.

Left margin

Use the *Left margin* prompt to set your Form's left margin. By default, Advanced Revelation starts the left margin five spaces from the left edge of the page.

For example, suppose you want a two-inch left margin for writing notes and comments on your report. If your printer prints ten characters per inch, you would enter 20 at the *Left margin* prompt.

Form heading

Use the *Form heading* prompt to print a heading at the top of each page. Enter the text of the heading exactly as you want it printed on your Form.

You can include options to insert a value such as the current date or page number. For example, the heading definition for a CUSTOMER Form might be:

```
Customer Table as of 'D' Page 'P'
```

This heading definition causes Advanced Revelation to insert the date and page number on your Form.

Form footing

Use the *Form footing* prompt to print a footing at the bottom of each page of your Form. Enter the text of the footing exactly as you want it printed on your Form.

As with headings, you can include options in your footing when you want Advanced Revelation to insert a value in your footing. For example, this footing definition

```
Time of printing: 'T'
```

causes Advanced Revelation to insert the current time and date in the footing.

To use an option, enclose it in single quotes at the place in your text where you want Advanced Revelation to insert the value. For a complete description of each HEADING and FOOTING option, see the R/LIST section of the *Reference* book.

Sending a Form to the printer or the screen

Use the *Option* prompt to direct your Form to a terminal (computer screen) or a printer. By default, Advanced Revelation directs the Form to a terminal. Valid *Option* entries are:

P	Printer
T	Terminal

Testing a Form

The Test Run feature allows you to see how your Form looks. If the data table that the Form accesses already exists, you can use Test Run to print a test report using data from the table.

To test your Form, press [Shift-F1]. Advanced Revelation displays the Test Run Form window. At the first prompt, enter the number of rows you want to print in your test run. By default, Advanced Revelation prints 10 rows.

At the second prompt, enter "P" if you want to print your test on the printer. By default, Advanced Revelation displays the report on the screen (terminal). If you send the report to the printer, be sure your printer is on and ready to print.

When your entry is complete, press [F9] or click <Save>. Advanced Revelation then displays the report. When the printing is complete, Advanced Revelation returns you to your Form definition in Paint.

Printing Forms

To print your Form:

1. Choose **Run-Report-Form** from the Development menu. The Form window displays.
2. At the *Table name* prompt, enter the name of the table that holds the Form report definition. By default, Advanced Revelation stores Form definitions in the REPORTS table. If your report definition is stored in another table, enter the name of that table. For a list of available tables, press [F2] or click <Options>.
3. At the *Form name* prompt, enter the name of the Form definition. For a list of available Forms, press [F2] or click <Options>.
4. Enter the keys for the rows you want included on the report. For a list of keys, press [F2] or click <Options>. To use a filter to select the rows printed on your report, press the Filter key ([Ctrl-F10]). For more information on creating and using filters, see "Using windows for queries". To print all rows, enter an asterisk (*).

17. Creating Labels

About labels

Use Labels for mailing labels or similarly formatted reports. You create labels using the Make Label window. To create or modify a Label report, choose the option **Define-Reports-Label** from the Development menu.

Creating a Label definition

Specifying a Label name and table

The first two prompts in the Make Label window are used to specify the name of the Label report and the table that contains the data displayed by the report. A description of these prompts follows.

Label definition name

Enter the name of the label definition you want to create or modify. Press [F2] or click <Options> to see a list of existing Label definitions.

Table name

Enter the name of the table that contains the data you want displayed on the labels. For a list of available tables, press [F2] or click <Options>.

Setting the Label size

Advanced Revelation provides a series of prompts to enable you to set the label size.

Lines on label

Enter the number of lines on your labels. For example, if your labels are one inch deep and your printer prints six lines per inch, you would enter 6 at this prompt.

Columns on label

Enter the number of columns on your labels. For example, if your labels are 3.5 inches wide and your printer prints ten spaces per inch, you would enter 35 at the prompt.

Sending Your Labels to the printer or the screen

Use the *Option* prompt to direct your labels to the terminal (computer screen) or the printer. By default, Advanced Revelation directs the labels to the printer. Valid entries are:

P	Printer
T	Terminal

Setting number of Labels to print across

Enter the number of labels you want to print across a page at the *Labels across page* prompt. By default, Advanced Revelation prints one label across a page.

Setting the spacing between labels

Tell Advanced Revelation how much space to leave between labels by indicating the horizontal and vertical spaces between your labels.

Horizontal

By default, Advanced Revelation leaves one blank line between labels. To change this setting, enter the correct number of lines at this prompt. For example, separate labels with three blank lines by entering 3.

Vertical

If you are printing more than one label across a page, Advanced Revelation leaves one blank column between the labels by default. Change the default by entering a number at this prompt. For example, if your labels were separated by two columns (or spaces), enter 2.

Defining your Label's layout

Use the bottom half of the Make Label window to define your label's layout. Place field flags where you want data displayed and enter any text you want printed on each label. To move between lines in the label layout area, use the arrow keys and the [Enter] key.

A field flag designates the name of a column from which data will be extracted. To add a field flag to the label layout, position the cursor where you want the column to appear, and type the column name enclosed in braces ({ }):

```
Attn: {CONTACT}
{COMPANY}
{ADDRESS}
{CITY}, {STATE}   {ZIP}
```

Field flags (in braces) indicate what columns to display in the label. Text outside the braces is printed literally.

For help adding field flags and field format specifications, you can press [F2] or click <Options> at the *Label layout* prompt. When the Field Flag popup displays, choose option 1, Columns (Dictionary), to select a field flag from a popup. Choose option 2, Format Specifiers, to add formatting specifications to the field flag.

Note For more information about your options at the Label layout prompt, press [F1]. For more information about format specifications, see "Field Flag Format Specifications" in the chapter "Creating Merge reports".

Testing a Label definition

To test your current label layout, press [Shift-F1]. Advanced Revelation displays the Test Run Label window.

At the first prompt, enter the number of rows you want to print in your test run. By default, Advanced Revelation prints 10 rows.

At the second prompt, enter "P" if you want Advanced Revelation to print your test on the printer. By default, Advanced Revelation displays the report on the screen (terminal). If you send the report to the printer, be sure your printer is on and ready to print.

When your entry is complete, press [F9] or click <Save>. Advanced Revelation then displays the report. When the printing is complete, Advanced Revelation returns you to your label definition in the Make Label window.

Printing Labels

You have the option of printing your labels directly from the Make Label window. Press [Shift-F1] to initiate a test run. To print the labels, save the label definition by entering "Y" at the "OK to save?" message window. The Test Run Label window appears. Enter the number of sample labels to print and direct the labels to the terminal (T) or printer (P). Press [F9] or click <Save> to print the labels. If you are printing on the printer, be sure your printer is on and ready to print.

To print your labels at any other time, choose **Run-Report-Label** from the Development menu.

Related topics

TCL LABEL command.

Using the Access Label window

A description of the prompts on the Access Label window follows. After you fill in the window, press [F9] or click <Save> to save your entry and print the labels.

Label name

Enter the name of the label definition. For a list of available labels, press [F2] or click <Options>.

Options

Use the *Options* prompt to direct your labels to the terminal (computer screen) or the printer. By default, Advanced Revelation directs the labels to the terminal. The valid entries are:

P	Printer
T	Terminal

To use a filter to select the rows printed on your labels, press the Filter key [Ctrl-F10]. For more information on creating and using filters, see the chapter "Using windows for queries".

18. Using EasyWriter

Introducing EasyWriter

This chapter explains how to create R/LIST reports and how to select, sort, and manage lists of rows using EasyWriter, a simple-to-use, popup-driven tool.

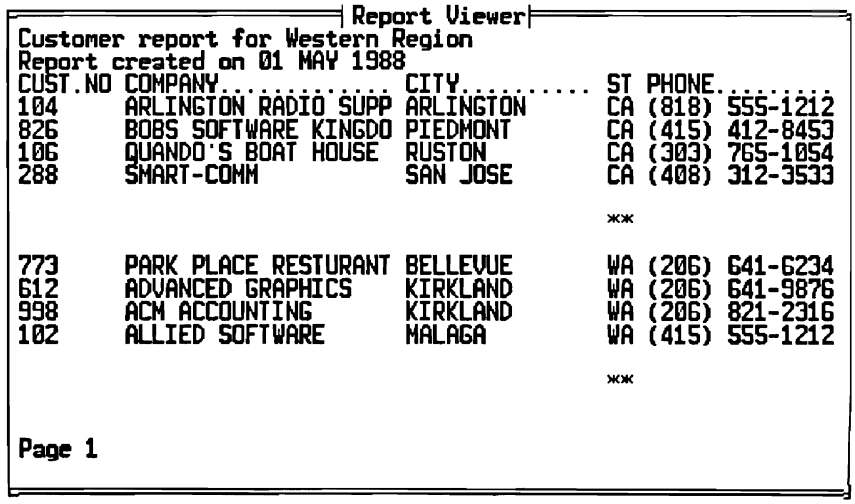
EasyWriter provides almost every capability inherent in R/LIST, yet frees you from having to remember the keywords and syntax of the report language. EasyWriter is especially useful to novice users since it guides them through the entire report writing process in a matter of minutes. Advanced users will also appreciate EasyWriter, because everything needed to create a report—from the table and dictionary names to the R/LIST keywords— is available by making choices from popup windows.

EasyWriter is more than a simple tool, however. Not only does it provide a convenient way for you to create reports, but it includes a variety of report management tools. EasyWriter can recall and re-execute any EasyWriter command you have recently entered. In addition, EasyWriter maintains a full report library, which permits you to create and store report commands you need often. Finally, EasyWriter can import R/LIST commands created earlier by you or other users, and modify, re-execute, and store them.

EasyWriter and R/LIST

EasyWriter actually builds and executes R/LIST commands. R/LIST is Advanced Revelation's ad-hoc retrieval language. Using EasyWriter and R/LIST, you can produce columnar reports from any table currently available.

For instance, you might create a report of all customers currently on file. When you create your report, it might look like this:



By harnessing the power of R/LIST, EasyWriter provides you with the capabilities you require to produce almost any report from an Advanced Revelation table. EasyWriter offers you these reporting options:

- Create a report from any table currently available to your system.
- Display any columns from the table in your report.
- Order (sort) the report in any way you like. You can sort the report by multiple columns (for example, first by state, and within like states, by city). In addition, you can specify that sorts be ascending (for instance, A to Z) or descending (Z to A).
- Select any rows out of the table to appear in the report. For example, you can specify that only those rows are to be displayed in which the state is "Florida," or those in which the invoice balance is greater than $1,000.
- Create custom headings and footings for your report. You can include the current date, time, page number and more in your heading.
- Create running totals and subtotals or averages for numeric data.
- Save your command to be used later. EasyWriter maintains a report library to store commands you need repeatedly.

Each of these capabilities is discussed in more detail later in this section.

EasyWriter and select lists

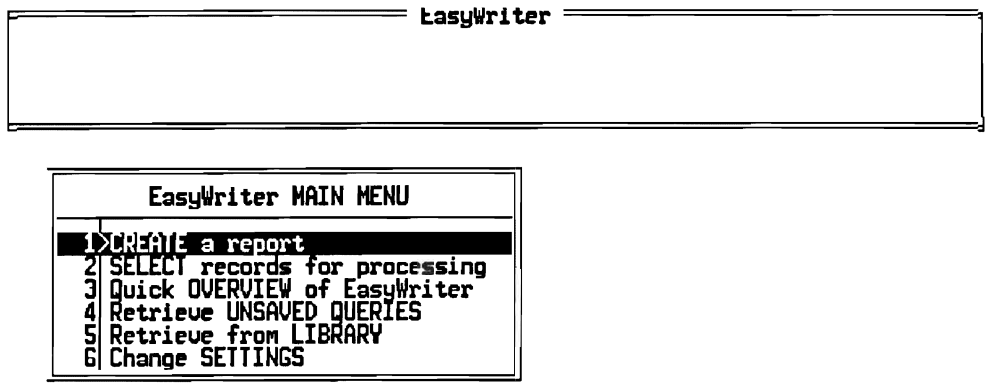
In addition to creating columnar reports with EasyWriter, you can create a select list—a list of rows to be used by another process. For example, you can use EasyWriter to create a list of all customers with a balance due greater than $100.

Not only can you create this list of rows using EasyWriter, but you can store it away as well. Later you can activate the list. The list can then serve as the basis for mailing labels, statements, or some other process, including an EasyWriter report.

Invoking EasyWriter

To call up EasyWriter choose **Define-Report-EasyWriter** from the main menu.

When EasyWriter starts up, you will see this menu and window:



On the left is the EasyWriter Main menu. At the top of the screen is the EasyWriter window. As you create a report, the EasyWriter window always displays the R/LIST command you are building. This enables you to see what you have created so far, and what effect your choices have on the command.

EasyWriter at the Filter and Select windows

At these windows you can use the [F2] key or click <Options> to call EasyWriter for help to write a SELECT or LIST command:

Window	Press [F2] or click <Options>
Filter	at *Selection* prompt
Select	at window
Make R/LIST	after entry at *Template name* prompt

Create the SELECT or LIST command at the EasyWriter window and then press [ESC] to return the command to the calling window (that is, the Filter, Select, or Make R/List window). When the current command displays at the calling window, press [F9] or click <Save> to run that window process.

The Report Library

EasyWriter maintains a report library of the LIST commands you have built. Anytime you build a LIST command, you can store it away in the library, to be recalled at another time.

This capability allows you to create a command once, and then store it away. Rather than recreating the command each time you require it, you can simply retrieve it from the library. You can edit and re-execute the command, and even store an edited version of it in the library. EasyWriter maintains the library of LIST commands as rows in the REPORTS table.

When you store a command in the Report Library, you do not store the full report itself. Instead, you store the R/LIST command that generated your report. When you retrieve and re-execute the command at another time, the report it creates is generated from scratch, using the most current data.

Getting help

Help is available throughout EasyWriter. Press [F1] (Context help) at any popup to learn more about your current options. For example, to learn more about the choice **Retrieve from LIBRARY**, press [F1] at the EasyWriter main menu. A message appears with information about each of the functions listed on the main menu.

Press [Ctrl-F2] (Concept help) to see a message that summarizes the process (selecting, sorting, etc.) you are currently constructing. Concept help is designed to give you an overview of the process, explaining its philosophy and purpose.

In addition to these help options, an explanation of the general functions of EasyWriter appears later in this chapter under the topic "Components of EasyWriter". This section explains the major components of the report writer such as displaying columns, selecting rows, etc., and how these functions relate to the popup windows that constitute EasyWriter.

Automatic help

The option **Set "EASY" MODE ON** from the Change SETTINGS popup enables you to set a Help level. If you set the Help level to Novice mode, explanatory messages will appear as you work through the components of EasyWriter. Each time you enter a new function, a message will explain the function to you, and remind you about keystrokes, etc., that are applicable there.

We recommend that you set Easy Mode ON until you are comfortable with EasyWriter. When you have experience with the report generator, you can set Easy Mode OFF so that the messages do not automatically appear. You will still be able to display them if you need them however, by pressing [Ctrl-F2] (Concept help).

Using popup windows

To use EasyWriter effectively, you will need to know how to use popup windows. EasyWriter uses four types of popups.

Single selection

This popup enables you to choose a single option and return. To use this type, highlight your choice, and then press [F9] or [Enter], or click <OK>.

Multiselection

This type enables you to choose one or more options and return. Highlight an option, and press [Enter] or click on it to tag it. Tag all the options you wish, then press [F9] or click <OK> to return the selections.

Sort

This type of popup allows you to reorder elements. For example, this allows you to rearrange the order in which columns appear left-to-right on the report. To use this type, highlight the element to move, and press [Enter] or click on it to tag it. Then move the highlight bar to the proper location in the popup, and press [Enter]. The tagged item will be moved to the location you highlighted. Press [F9] or click <OK> to save the new order of elements.

Toggle

This type of popup alternates or toggles between two choices. For instance, a toggle popup enables you to choose between ascending and descending sorts. To toggle an option, highlight it and press [Enter] or click on it. To toggle to the previous value, press [Enter] again. Press [F9] or click <OK> to save your specifications.

To highlight an option in a popup, use the cursor keys to move up and down. You can also enter the number of the choice you wish to highlight.

If you know the text of the option you wish to highlight, you can type in the first character(s) of the word, and then press [Enter]. For example, if you type in "Ret" at the EasyWriter main menu and press [Enter], the selection **Retrieve UNSAVED QUERIES** will be highlighted. If you do this, be attentive to what you type, because the search function is case-sensitive.

Anytime that you want to cancel your choice, or to back up in a series of popups, press [ESC]. This exits the current process without saving any changes, and returns you to where you were when you called the popup.

Components of EasyWriter

EasyWriter works by providing you with a series of popups and windows. These guide you through the process of constructing the R/LIST command for your report or filter. Each component of the command or filter—the selection (comparison) criteria, sort criteria, display criteria, headings, footings, and modifiers— is comprised of a series of these popups and windows. You simply need to choose the option(s) you want from the popups as they appear, and fill in data at the windows as prompted.

Remember that the online Help system will provide you with details about the popup or window you are using. Press [F1] (Context help) to see specifics about the popup

options that are available . Press [Ctrl-F2] (Concept help) to read more about the purpose of the process you are using.

The current report

For most EasyWriter processes, you will be creating, modifying, or saving a current report. For instance, the option **CREATE a Report** walks you through the process of creating an R/LIST command. As soon as you choose this option from the EasyWriter main menu, you will begin constructing a new report. That new report will then be the current report.

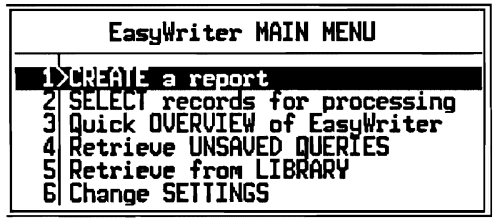
On the other hand, if you choose the option **Retrieve from LIBRARY**, you will be recalling a report created earlier. As soon as you select an item from the Report Library, that item becomes the current report.

When you are at the EasyWriter main menu, there is no current report. Because of this, there is no option at the main menu to add a report to the library, or to modify a report. In order to access these functions, you must first retrieve or create a report to serve as the current report.

Conversely, if you are editing the current report, and attempt to return to the main menu, you will see a message warning you that if you return to the main menu the current report will be cleared. If you proceed, the current report is cleared and you have the opportunity from the main menu of creating or retrieving a different report.

The main menu

The EasyWriter main menu is the entry point into the six basic functions you can access. The main menu looks like this:



CREATE a Report

When you choose this option, the end result will be a columnar report that displays on the screen or is printed on the printer. You will have the opportunity to specify six components of the report:

- Which table you wish to report on
- The columns that are to be displayed
- The order in which the rows are to appear
- Which rows are to appear
- Headings, footings, and other formatting specifications, including subtotals and averages
- Whether the report displays on the screen or prints to the printer

The word LIST will appear in the EasyWriter window, as will a popup displaying all the tables available to EasyWriter. You must select the table for which you are creating a report. Once you have done so, you may construct the rest of the components of the command using popups.

SELECT Rows

The result of this option is not a report, but a list of row keys. Use this option when you wish to set aside a list of rows to be used in another process.

For example, you might have a process that prints mailing labels. You can use the SELECT option to create a list of all customers in a specific area of the country. You can save the list, and then later retrieve the list and use the Advanced Revelation Label processor to create mailing labels.

The **SELECT rows** option, like the **CREATE a Report** option, permits you to specify which rows to select, and in what order. Because you are not creating a report, you do not need to specify display columns or formatting options, or to indicate screen/printer output.

When you first choose this option, the word SELECT appears in the EasyWriter window. You will also see a popup of available tables. You must choose the table from which you will be selecting rows. Once you have done so, you can specify the rest of the options (which rows, what order) by using the popups.

UNSAVED Queries

This option is used to retrieve and re-execute the most recent reports or row key selections. You can use this option even if you did not save your report command in the Report Library.

Advanced Revelation always remembers a certain number of your *most recent* commands (the exact number depends on your user environment). EasyWriter can look through these recent commands and pick out the ones that generated a report or a row key list.

Retrieve from LIBRARY

Anytime that you create a report or a selection command, you can store the command in the Report Library. This option from the EasyWriter main menu allows you to retrieve stored commands.

In order to store the command in the Report Library, you must use the option **ADD report to LIBRARY**. This option appears on both the REPORT options and the SELECT options popups. To use the option, first create the report or selection command, and then choose the **ADD report to LIBRARY** option to store the command.

Change SETTINGS

EasyWriter can be configured to suit your environment, your application, and your level of expertise.

Environmental settings include the size of the screen and printer page. For example, if your printer can only print 80 columns across a page, you can use this option to make EasyWriter aware of this.

Finally, you can set a Help level for EasyWriter. If Easy Mode is set to ON, EasyWriter will display help messages as you move from popup to popup. The messages inform you about the popups you are using, and remind you about important keystrokes. When you are familiar with EasyWriter, you can set Easy Mode to OFF so that these messages do not appear automatically.

Specifying WHICH ROWS

This option, which is called from both the CREATE a Report process as well as the SELECT Rows process, is used to tell EasyWriter what rows you wish to use in your report or in your row list. In order to do this, you must create one or more comparison clauses.

The comparison clause

A comparison clause determines which rows will appear in the report. The clause compares the value of a column against a fixed value or against the value of a different column in the row. If the comparison for the current row results in a match, the row is included in the report. If not, the row is not displayed.

For example, you may wish to print a report displaying all customers in the state of California. The comparison clause will then look into the state column of each row. If CA appears in the column, the row is displayed.

A comparison clause consists of at least these elements:

- A column to base the comparison on
- A comparison operator, such as = (equals), > (greater than), etc.
- A value or column to compare against the data in the original column

To use the above example, the comparison clause to select California customers might read something like:

```
ST EQ "CA"
```

The column to base the comparison on is ST (state). The comparison operator is EQ (equal), because you want to find rows in which the state is (equals) California. Finally, the value to compare to is CA.

Creating a comparison clause

To create a comparison clause, you will step through a series of popups. Assume for a moment that you are creating a report based on the CUSTOMER table. The sequence of popups will be:

CUSTOMER Columns (choose the original column)
⇓
COMPARE Words (choose the operator =, >, etc.)
⇓
CUSTOMER Columns (choose the comparison word or column)

From these popups, you can construct the comparison clause ST = "CA" or something similar.

Multiple comparisons (ADD a COMPARISON)

You can expand the comparison clause to compare the contents of a column against more than one value. For example, perhaps your report is to be for customers in California and in Nevada. In that case, the comparison clause would read:

```
ST = "CA" OR EQ "NV"
```

This causes the report to select all rows in which the state column contains either "CA" or "NV"—either is acceptable.

Change the relationship between comparisons

When you add an additional comparison value, it is connected to the earlier values with an OR. This means that if any one of the values in the comparison clause appears in the column, the row is displayed.

There will be instances in which this is not appropriate. For example, you may wish to find all customers in which the company name contains the words ACME and INTERNATIONAL. In this case, both words have to appear. If only one of these words appears in the company name, the row is not acceptable.

To create this type of comparison clause, create a clause with multiple comparison values. Then change the relationship between the comparisons. The option **Change RELATIONSHIPS between comparisons** allows you to change the word OR to AND. This changes the comparison so that all the values in it must appear in the column.

Multiple comparison clauses

Sometimes the rules for displaying rows involve more than one column. For example, you may wish to create a report listing all 1988 invoices for your Florida customers. The comparison clauses to generate this report examine two columns: ST and INVOICE_DATE.

If you examine more than one column, you must create more than one comparison clause. In this instance, you create the first comparison clause based on the ST column:

```
ST EQ "FL"
```

A second comparison clause is based on the INVOICE_DATE column:

```
INVOICE_DATE FROM "01-01-88" TO "12-31-88"
```

The default relationship between multiple comparison clauses is that either criterion is acceptable (an OR is placed between the clauses). In this example, however, because both criteria must be met for a row to be displayed, you must change the relationship between the separate clauses so that there is an AND between them.

There is no limit to the complexity of the rules for displaying rows. You can have any number of clauses, each of which examines any number of values. You can establish any relationship you require using AND and OR to connect the values or clauses.

Including vs. excluding rows

When a comparison clause has been established, a report compares the contents of a column against a fixed value or against another column. If the row fulfills the criterion in the comparison clause, it is included in the report.

You can also create comparison clauses that cause rows to be excluded from the report. For example, you might create a report that displays all customers except those whose status is "Inactive".

You can use the **Change comparisons to INCLUDE/ EXCLUDE rows** option of the COMPARISON Options popup to change between inclusive and exclusive comparison clauses. If a clause is inclusive, it begins with the word "WITH". An example is:

```
WITH ST EQ "CA"
```

An exclusive clause begins with the word "WITHOUT". An example is:

```
WITHOUT STATUS = "Inactive"
```

Selecting Specific Rows

If you know the row key or keys of the rows you wish to display, you can use the **CHOOSE rows using SPECIFIC KEYS** option from the COMPARISON Options popup to enter these. This will result in a faster report than if you base your comparison on a non-key column.

For example, if you are creating a report of certain employees, you could base the report on the employees' names. However, the report will run faster if you base it instead on the column you use to identify the employees in the table (the key column). This might be the employees' social security numbers or their employee numbers.

Specifying row sort order

Rows in a table are not maintained in any particular order. If you wish to display them in a specific order, you must order (sort) them in the report.

The option **Specify row SORT ORDER** allows you to do this. Sort order is based on the contents of columns in the row. For example, a report of customers might be sorted by the contents of the ZIP column, the COMPANY column, or some other column.

You can sort by any number of columns. If you specify a second sort column, it will be applicable to rows that have the first sort column in common. For example, if your first sort is by ZIP code, and your second one by company, the report will appear in ZIP code order. If there is more than one row with the same ZIP code, that group of rows will appear in company order. Third, fourth, etc. sort columns are applied in a similar hierarchy.

Ascending and descending sorts

Unless you specify otherwise, sorts are always ascending. Rows are sorted from lowest to highest value. Rows with character data will appear in alphabetic order, and rows with numeric data in numeric order.

If you need to, you can change one or more sort columns to use descending order. For example, if you want a report with the most recent date first, you can specify a descending order for the date column.

Displaying rows

In addition to simply displaying a column in a report, you have several *options for* modifying the display. These include reformatting a column, totaling or averaging a numeric column, and grouping columns that have been sorted.

Reformatting a column

When you display a column in a report, the display characteristics of that column are read from the dictionary of the current table. These characteristics include the display length, the justification of the column (left, right, center or text (paragraph)), the column heading, and the display type (conversion). If you reformat a column, the formatting specifications are in effect only for the current report.

For example, you might decide that you would like to display the COMPANY_NAME column. As defined in the dictionary, it might be 25 characters wide. However, you decide that since you don't have much room on the report, you can get by with only 20 characters of the company name.

To temporarily change the display length of the COMPANY_NAME column, choose the **REFORMAT a display column** option from the DISPLAY Options popup. This displays a window with prompts for the four format specifications you can change. Choose the column to reformat, and the window will appear with the existing (default) values for these specifications filled in. You can change any of the four values. Use the [F2] key or click <Options> to see options for *Justification* and *output format*.

Total and average

If you have columns with numeric data in them, you can have a total or an average of those columns printed at the end of the report. Numeric columns are those defined in the dictionary with a display type of numeric (MD).

If you also group the columns you are totaling or averaging, you can get a subtotal or subaverage printed at each break in the report. A final total or average then appears at the end of the report.

Grouping rows

If you are sorting (ordering) your report and displaying the columns on which you are basing the sort, you can choose to group those rows. When you group rows, the report shows a "break" when the value of the sort column changes.

For example, if you create a customer list sorted by zip code, you might also group the report by zip code. Whenever a new zip code displays on the report, a break in the report—some blank lines, or even a new page— is generated in the report. Because grouping causes a break in the report whenever a new value is encountered, it is also called "breaking on" a value.

The normal break in a report is a blank line, a break indicator, and then another blank line. The break indicator is normally *** (three asterisks). Options available from the GROUP (BREAK-ON) ordered values popup allow you to substitute different break indicators.

Alternatives to the three stars include starting a new page, printing the value of the group column, or including text that you supply. If you are also totaling the column, a subtotal will appear at each break. You can specify options that display an underline and an overline for the subtotal. You can even have the value of the group column appear in the heading—this is useful if you are also specifying that each new group be put on a separate page. If you choose this option, you must choose a similar option for the heading.

Heading and footing

A heading is text that appears at the top of every page of the report. A footing appears at the bottom of each page.

A heading or footing consists of text that you provide. Options in the HEADING window and in the FOOTING window enable you to include additional information. For example, you can display the current date, time, page number and table name in addition to your own text. Press [F2] or click <Options> at the HEADING and FOOTING windows to see a list of these options.

You can also include the value of a column for which you have specified the group option. To do so, you must specify this same option in the GROUP (BREAK-ON) options popup.

Other formatting options

Whenever EasyWriter creates a report, it makes certain assumptions about the format of that report. Unless you specify otherwise, the report will always:

- Display a heading
- Display the row key
- Display column headings
- Display all detail rows when generating totals
- Single-space the report

The Special FORMATTING options popup enables you to change these defaults. For example, you can specify that detail rows are not to be displayed when you are generating subtotals. If you do so, the report will consist of subtotals and a total only—the individual rows from which the totals are calculated will not display.

Printer and screen

Reports are normally displayed on the screen. If you wish to route a report to the printer, you must choose the **Reroute to PRINTER** option from the REPORT options popup.

Once you have chosen this option, the popup changes. The option will change to read **Reroute to SCREEN**. This permits you to change your mind and reroute the report back to the screen.

ACTIVE KEY List

Lists of active keys are typically used when you wish to select and sort a set of rows, and then use that list of rows several times. If you generate the list once and reuse it many times, you can avoid selecting and sorting the rows each time you need them.

To create a list of active keys, use the **SELECT Rows for processing** option from the EasyWriter main menu. This enables you to select and sort a list of rows. When the list is assembled you can use the **Change ACTIVE KEY list** option to save the keys away.

The same option enables you to retrieve the keys later. When you do so, the list of active keys becomes the basis for further processing. For example, if you create a report, and then activate a list of keys representing a subset of a table, the report will only display rows whose keys are on the list.

The advantage is that you do not need to select and sort rows each time you run the report. You can select and sort the rows one time, and then store the list of rows away. You can then activate the list whenever you like, and run a report against it. The report does not need to include selection and sort criteria, since these have already been incorporated into the list of active keys.

If you activate a list of keys, but then decide that you do not need that subset of rows, you must clear the list of active keys. An option from the Change ACTIVE KEYS options popup enables you to do this.

Importing R/LIST commands

EasyWriter can import almost any LIST or SELECT command already available. Commands are imported into EasyWriter by executing them and then retrieving them (**Retrieve UNSAVED QUERIES** option). You can also import a command by placing it into the REPORTS table directly, and then using the **Retrieve from LIBRARY** option. Commands imported in this manner can be edited, re-executed, and stored by EasyWriter.

There are certain features of R/LIST, however, that EasyWriter cannot use. If

EasyWriter encounters one of these while importing a command, a message appears stating this fact, and the import process is halted.

R/LIST features that EasyWriter cannot use are:

- NOT
- EVERY
- LIMIT
- Parentheses
- DICT override

19. Creating and using Merge reports

Introducing Merge

Merge provides a full-featured, flexible method of merging data from an Advanced Revelation table into a form or document that also includes literal text, variable information such as date and time, headings and footings, and even special text such as personalized or conditional messages.

In addition to its basic capabilities, Merge includes a variety of advanced features. These include individual paragraph formatting to create left justified, filled and unformatted paragraphs, and multipage capabilities such as widow and orphan control and control for variable-length tables. Character formatting includes underlining, boldface, and other desirable features.

Finally, Merge is able to accommodate unusual or complex requirements. Portions of a Merge template can be made conditional, and many points in Merge are open-ended to accept developer customization.

Merge vs. Form

Advanced Revelation already includes a forms processing capability in Form. However, there is a difference in the use of these two reporting tools.

Form is suited to creating forms that you can paint out on the screen. This is particularly useful in situations where you wish to merge data from an Advanced Revelation table with a preprinted form or similar fixed format. In general, Form is used to produce output that looks exactly like the template on the screen.

Merge, on the other hand, is better suited to creating forms for variable data. Unlike Form, Merge is expressly designed to accept and format varying quantities of data. This includes the ability to merge and then format data into paragraphs, and to flow text and data onto multiple pages if necessary.

Fundamentals of Merge

Merge works something like Label. You create a Merge template that includes text for the body, data from a table, plus headings, footings and control information such as data table name, page size, margins, etc. To do this, you fill in prompts in a special window designed to help you create Merge templates.

When the Merge template is done, you run a command that reads the template and merges data from the data table into the template. The result is a series of finished documents or reports that can be displayed on the screen, printed on a printer, or written to a table.

The Merge template consists of the control information for the document as a whole, plus the basic text, called the *script*. Variable information such as data from the table

or character or paragraph control information is included in the text using special sequences called flags. For example, the following text illustrates the use of a flag to include data from a column in the data table:

```
Dear {CONTACT_NAME}:
```

In this example, data from the column CONTACT_NAME will be inserted into the text at this point.

Control information is embedded in a similar fashion. The following illustrates the use of flags to cause portions of the text to be put into italics:

```
send your payment «ITAL»today«END.ITAL»!
```

In the example, the word "today" will appear italicized in the final output.

Creating a Merge template

The first step is to create a Merge template. The Merge template contains information such as the basic format of the document and the script itself.

The Make Merge window

To create a Merge template, use the Make Merge window. The Make Merge window is available by using the **Define-Report-Merge** option from the Development menu. The prompts in the window are grouped by function. For example, all prompts concerning margins appear together under the label MARGINS.

Fill in the prompts in the window to create the Merge template. Then press [F9] or click <Save> to save the template in the REPORTS table. Be sure to use [F1] if you are unclear about the purpose of a prompt. Many prompts also provide options if you press the [F2] key or click <Options>.

Make Merge window menu

The Make Merge window has its own menu, available when you press [F10]. The menu includes options to test scripts, do a test run of the script, and import scripts from an operating system file. The test options on this menu are discussed later in this chapter under "Running a Merge report". Other features of the menu are described under "Special features of Merge".

Make Merge window prompts

The prompts in the Make Merge window enable you to establish the Merge script itself, as well as to enter a variety of control information about the Merge template.

Merge name

Whenever you create a Merge template, you give it a unique name. The template is filed away using this name, and you can recall it anytime later. The Merge template name can be up to 50 characters long and can include alphanumeric characters, $ (dollar sign), and _ (underscore). To see a list of existing templates, press [F2] or click <Options>.

Data

Enter at this prompt the name of the table containing the data you wish to merge with the template. For example, if you wish to merge this Merge template with data from the CUSTOMER table, enter CUSTOMER at this prompt. Press [F2] or click <Options> to see a popup of all available tables.

Special text

If you will want to merge special text such as personalized notes or postscripts with the document, enter here the name of the table that contains that text. Press [F2] or click <Options> to see a list of tables available.

For more information about the use of Special Text, see "Special features of Merge" later in this chapter.

Output

If you enter the name of a table at this prompt, Merge will route its output to this table. This is used to create an entire table of merged documents before to printing them. For more information on the use of the output table, see "Running a Merge report" later in this chapter.

Top

This is the number of rows left blank at the top of the page. For example, to leave a one-inch margin at the top using a standard 6 lines-per-inch printer, enter 6 at this prompt.

The heading text (see below) is printed in the top margin, so be sure to provide enough space for the heading you create.

Bottom

Enter at this prompt the number of blank lines to be left at the bottom of the page. For example, to leave a one-half-inch margin at the bottom of the page using a standard 6 lines-per-inch printer, enter 3 at this prompt.

The Footing text (see below) is printed in the bottom margin, so make sure that you are allocating enough space for it.

Left

This is the number of columns to be left blank on the left side of the page. This number represents the number of characters in the current pitch (10 pitch = 10 characters per inch).

For example, if you wish to leave one inch blank on the left side of the page, and your printer is set to print 10 characters per inch, enter 10 at this prompt.

Right

This is the number of columns that will be left blank on the right side of the page. This number is calculated in the same manner as the left margin.

The right margin represents the minimum amount of space left blank on the right side of the page. If you format paragraphs using left justification, or no justification at all, text will appear "right-ragged," with each line ending at a different point in the page. None of the lines, however, will run farther than the right margin.

Para indent

The number entered at this prompt is used as the default number of columns by which the first line of each paragraph is indented. For example, if you enter 5 at this prompt, the first line of each paragraph in the document will be indented five characters.

This number can be positive or negative. If the number is positive, an indented paragraph is created. If the number is negative, the first line of the paragraph is flush with the left margin, and the remainder of the paragraph is indented. For example, if you enter -5 at this prompt, all lines in a paragraph except the first one will be indented five lines from the left margin.

Widow

This prompt controls the minimum size of a paragraph that will appear at the bottom of a page. This feature is used to avoid having lines from paragraphs "widowed" when the bulk of a paragraph appears on the next page.

The number entered at this prompt defines the size of a "widow". Anytime that a paragraph splits between pages and leaves this many lines or fewer, it will be reformatted to fit onto the next page.

Orphan

An "orphan" is the opposite of the widow described just above. An orphan appears at the top of a page.

The number of lines entered at this prompt defines an orphan. If a paragraph splits and flows this many or fewer lines to the next page, it will be reformatted to fit onto the next page.

Width

This number defines the total size of the page in columns. For example, a typical letter-sized piece of paper is 8-1/2" wide. You would therefore enter 85 at this prompt.

The number entered at the *Width* prompt is used in conjunction with the left and right margins to establish the size of the area in which text can appear. For instance, a page 85 columns wide less a 10-column left margin and a 10-column right margin leaves 65 columns in which to print text.

Height

The number entered here is the total length of the page in rows. For example, a typical letter-sized page is 66 lines long.

This number, less the space for the top and bottom margins, determines the size of the area in which the body of the text will appear.

Default format

Each paragraph in the document can have its own format. However, the entry at this prompt determines the format for paragraphs having no special format.

Merge recognizes three types of paragraph formatting (press [F2] or click <Options> to see a popup of formatting options):

LEFT	The left margin is aligned, but the right margin is ragged.
FILLED	The paragraph is padded with spaces so that each line is aligned at both the left and right margins.
NOFORMAT	No attempt is made to format, and text is output as it appears in the script, except for formatting changes.

Header

Use this prompt to enter text that will appear at the top of each page of the document. The heading can be as long as the top margin. Be sure that you have allocated enough space in the top margin to accommodate the heading.

To enter text for the heading, begin typing at the prompt, or press [F2] or click <Options>. In either case, a Merge Header window will display, allowing you to enter or edit the heading text.

Enter text as you would in any window. The heading text is treated as a normal script, so all page and paragraph formatting is applicable. You can also include all types of flags, such as data columns or flags for paragraph or character formatting.

For more information on column, paragraph and character flags, see "Entering scripts" later in this chapter.

When you are done entering heading text, press [F9] or click <Save> to save the heading. Press [Esc] to return to the Make Merge window.

Footer

Use this prompt to define up to three lines of text that appear at the bottom of each page of the document. You can enter and edit footing text exactly like heading text (described just above).

Merge script

This prompt appears by itself on the second page of the Make Merge window. It is used to enter the body text of the document.

As with the heading and footing, entering or editing the script is done in a separate window. When you begin typing, or if you press [F2] or click <Options>, the Merge Script window will appear. All script entry and editing is done in this window.

Enter the text of the script in the Merge Script window. You can embed flags in the script in order to control paragraph and page formatting. For more information on the use of flags, see "Entering scripts" later in this chapter.

When you are through entering or editing your script, press [F9] or click <Save> to save it. Then press [Esc] to return to the Make Merge window.

In the Make Merge window, press [F9] or click <Save> to save your Merge template.

Entering scripts

The script is the heart of the Merge document. It contains the text that will appear when the document is merged with data from a table. In addition, it contains information about what data will appear in the document, as well as the format of paragraphs and characters in the final document.

The script consists of literal text, which is to appear exactly as you enter it, except for format changes. The script also contains flags for indicating data columns to merge, as well as flags that specify paragraph, document, and character formats that are to be used in the text. These flags are translated into the appropriate data or codes when the document is actually merged.

Scripts are entered at the *Header*, *Footer*, and *Merge Script* prompts in the Make Merge window. To enter scripts at these prompts, use the cursor keys or the [Enter] key to get to the appropriate prompt, and then press [F2] or click <Options>. This will display a special script entry window. Enter the script at this window. When the script is finished for that prompt, press [F9] or click <Options> to save it, then press [Esc] to return to the Make Merge window. Press [F9] or click <Save> at the Make Merge window to save your Merge template.

While you are editing a script, the [F2] key (or <Options> option) also provides a popup of all flag options. From this popup you can in turn display other popups of all column, format, and character flags. The popups display the flag itself as well as a brief explanation of the flag's meaning. Anytime you wish to insert a flag, press [F2] or click <Options>. Special softkeys enable you to call the appropriate popup (column flags, character flags, etc.) directly.

Entering text

When the Header, Footer or Merge Script entry window displays, you can enter the script as if you were creating a document using a word processor. Use the cursor keys and the [PgUp] and [PgDn] keys to move around in the script entry window.

Note The script entry windows use the Advanced Revelation full-screen editor. For more information about using this editor, see the chapter "Using the full screen editor".

Enter text as you wish to see it displayed. Except for formatting text (see below), everything you enter will be displayed. Create separate paragraphs of text by placing a blank line between them.

For example, you might enter two paragraphs like this:

```
To take advantage of this offer, you must act today.

Return the enclosed coupon with your check, money
order or credit card number.
```

The text will appear exactly as you have typed it, except for any formatting you have specified. For instance, if your default format specifies justified text, the paragraphs will appear justified when they are printed or displayed.

Entering column flags [Alt-F3]

To incorporate data from a data table, you will need to include column flags in the script. Column flags designate the name of the column from which data will be extracted.

Column names

To include a column flag, press [Alt-F3] while in a script. A popup will appear that displays all columns from the data table for which you are creating the script. Choose the column that is to appear in the script at this point.

For example, to create a customized salutation in a letter, type the word "Dear", and then press [Alt-F3]. A popup of all columns appears. Select the column that is to appear in the salutation (such as FIRST_NAME). When you press [F9] or click <Save> to save your selection, you will be asked to select a column format. Skip this popup by pressing [Esc]. The following will appear in the script:

```
Dear {FIRST_NAME}
```

The braces ({ }) around the column name indicate that the text inside is the name of a column. When the document is merged, the first name data from the current row replaces the column flag.

Reserved words

The first four options in the Merge Columns Select popup are for special reserved words. These reserved words allow you to insert flags for the current date, time, and page number into the script.

Reserved word flags are treated like column flags. However, the words inside the braces (for instance, MRG.PAGE) are not replaced by data from the row. Instead, the special function called by that word (for example, the current page number) is merged at that point.

The reserved words used as column flags are:

MRG.PAGE	Insert current page number
MRG.TIMEDATE	Insert current time and date
MRG.DATE	Insert current date (internal format)
MRG.TIME	Insert current time (internal format)

Note If you use the flags for date or time (MRG.DATE and MRG.TIME), you will need to format the columns. For details on column formatting, see "Column formatting" later in this chapter.

Entering format flags [Alt-F1]

A special set of flags lets you indicate various types of paragraph and document formatting. Formatting flags are not surrounded with braces, as are column flags. Instead, they are surrounded with chevrons (« »). To see a list of available format flags, press [Alt-F1]while editing a script.

Formatting flags are divided up into three different types. The first type establishes or changes the format of paragraphs or lines. The second type formats a block of multivalued columns. The third type indicates conditional inclusion of text or other script information.

Paragraph formatting flags

Paragraph formatting flags are used to create or override a format for paragraphs or individual lines. Each of the flags has a different scope. Some are document oriented (they work for the document as a whole), while others are limited in scope to a single paragraph or line. See the following table.

Format Flags for Merge

Flag	Meaning	Scope
«UNFORMAT»	Turn default format off	Document
«FORMAT»	Re-establish default format	Document
«PARA.INDENT, *n* »	Establish paragraph indent of *n* characters	Document
«CENTER»	Center an unformatted paragraph	Paragraph

(cont)

Flag	Meaning	Scope
«END.CENTER»	End centering, effective for the current line	Paragraph
«DBLSPC»	Double space paragraph	Paragraph
«RIGHT»	Right-justify remainder of line, if unformatted paragraph	Line
«PAGE»	Force a page break	N/A
«PARA»	Force a paragraph break, no space	N/A

If the scope of a flag is "Document," the flag is in effect for the rest of the document unless explicitly turned off. For instance, once the «UNFORMAT» flag is activated, all subsequent paragraphs are unformatted until the «FORMAT» flag is encountered. Flags with a "Paragraph" scope only affect the current paragraph. Line-oriented flags only affect the current line.

A flag such as «RIGHT» or «CENTER» is only effective for an unformatted paragraph. This is because the paragraph format otherwise conflicts with this flag. If there is a conflict between a paragraph format and a flag, the paragraph format takes precedence.

The «PARA» flag is used to create paragraphs with no blank lines between them. Normally, a paragraph boundary is established by putting a blank line between two paragraphs. The «PARA» flag, however, forces a paragraph break without inserting a blank line. Inserting the «UNFORMAT» or «FORMAT» flags into a paragraph also forces a new paragraph at that point.

Multivalued blocks

A special set of flags enables you to define blocks of multivalued columns. Multivalued blocks are used to establish formatting and headings for groups of multivalued columns.

A multivalued block is marked with the «VBLOCK» and «END.BLOCK» flags . Inside of this block can appear column flags, literal text, or other standard script information. The information in the block will be repeated once for each value or set of values in the multivalued column. If the columns in the block have different numbers of values, the information is repeated for the maximum number of values in the longest column.

The block is repeated as a whole each time. All text and flags that appear between the start and end of the block — including spacing, carriage returns, etc.— will be repeated each time.

A multivalued block is treated as a normal paragraph, and is subject to any formatting

that is applicable to other paragraphs. As a result, spacing, carriage returns, etc., are all formatted as they would be in the current paragraph format.

In order to establish columns, a multivalued block *must* be tagged as an unformatted paragraph. If you have a default paragraph established, you should set the «UNFORMAT» flag before the multivalued block, and reset the «FORMAT» flag afterward.

Note Do not follow a «VBLOCK» flag with a blank line.

To print values in columns, the «VBLOCK» and «END.BLOCK» flags must be on separate lines. Either of these will print correctly:

```
«VBLOCK»
{column}{column}«END.BLOCK»
```

```
«VBLOCK»{column}{column}
«END.BLOCK»
```

Multivalued block headings

You can also designate a multivalued block heading. This heading appears at the top of the multivalued block, and is typically used as a column heading above the multivalued column. If the multivalued block is longer than one page, the heading is repeated on subsequent pages.

To establish a multivalued block heading, use the flag «VBLOCK.HEAD». Everything that appears after this flag and before the flag will be used as a heading. As with the block itself, all paragraph formats are applicable.

An example block that includes a heading might look something like this:

```
«UNFORMAT»
   «VBLOCK.HEAD»Invoice  Date  Amount
   «VBLOCK»  {INVOICES}  {INV_DATE}  {INV_AMT}
   «END.BLOCK»
   «FORMAT»
```

In the example, the multivalued block is enclosed with flags that first disable, and then re-establish paragraph formatting. A block heading is established with a fixed number of spaces between the columns. The block itself consists of three column flags with spaces between them. A carriage return follows the block, so that a return will be added for each value printed.

Conditional scripts

It is possible to designate the inclusion of text or flags conditionally. For instance, you may wish to display certain columns only if they exist, or only if other columns in the row exists.

The flags «IF», «ELSE» and «ENDIF» are used for conditional blocks. «IF» and «ENDIF» are required; «ELSE» is optional for situations in which you wish to designate alternative text and flags.

The condition is based on whether data appears in one or more columns. The columns to be tested appear in the conditional block as column flags. The column flags can appear anywhere between the «IF» and the «ELSE» or «ENDIF». For instance, the following illustrates the use of a conditional block:

```
«IF»By the way, you are eligible for a discount of
   {DISCOUNT} on anything in our catalog.«ENDIF»
```

In the example, if there is anything in the DISCOUNT column for the current row, the entire block will be printed (including the column DISCOUNT). If there is nothing in the DISCOUNT column, this block of text will be skipped.

An example of an «ELSE» flag:

```
«IF»Dear {FIRST_NAME}:«ELSE»To whom it may
   concern:«ENDIF»
```

Here, if a first name exists, the salutation will be "Dear" plus the first name. If no first name is on file, the text "To whom it may concern" is printed.

You can designate that a column is to be used as a condition, but that it is not to be printed. To do this, put "?" (question mark) as the first character in the column flag. For example:

```
«IF»{?INVOICES}We note that you have already placed an
   order with us.«ENDIF»
```

In the example, if there are any invoices on file for this row, the text is printed. Because the column flag includes "?", it will not be printed; it is only used to test the condition.

Finally, you can base a conditional block on a combination of column flags. If more than one column flag appears in the «IF» block, there must be data in each one of the columns in order for the block to be printed. Multiple columns in the «IF» block are linked with a logical AND.

If you wish to identify one or more column flags as particularly important, you can put * (asterisk, star) at the beginning of the column flag. This serves two functions. The first is that it causes only the column(s) so marked to be considered in testing the condition. All columns in the block are displayed, but unmarked columns are not included in testing the condition. For example:

```
«IF»Dear {*FIRST_NAME} {LAST_NAME}:«ELSE» To whom it
   may concern:«ENDIF»
```

In the example, if a first name appears, the "Dear" section of the block is printed. If a last name exists, it will be printed as well, but it is not used as a test for this condition. If no FIRST_NAME is on file for the current row (even if there is a last name), the

second half of the block is printed.

The * also changes the logic of testing. If * is used in column flags inside a conditional block, the columns are linked with a logical OR. If any of the marked columns contain data, the block is printed.

Entering Character Flags [Alt-F2]

Character format flags enable you to specify such formats as boldface, italic, underline, or other font characteristics. You can establish any character formats that your printer is capable of producing. To see a popup of available character flags, press [Alt-F2] while editing a script.

Character format flags look much like paragraph format flags. They consist of a keyword (such as ITAL for italic) surrounded by chevrons.

Character formats always come in sets—a start format flag and an end format flag. The start format flag establishes the character format. That character format is in force until the end format character flag is encountered, or until the document ends.

For example, a set of character format flags might look like this:

```
The book «ULINE»Understanding Radio
   Technology«END.ULINE» was first printed in 1952.
```

In this example, everything between the two flags—namely, the title of the book—will be underlined.

Remember to insert the end format flag at the point in the script where the character formatting is to end.

Note The special flags «FONT1», «FONT2», and «FONT3» refer to fonts defined for your printer. See the chapter "Configuring the workstation".

Column Flag Format Specifications [Alt-F4]

Anytime you include a column flag in the script, the data will appear in the script exactly as it exists in the row. For example, if data is all upper case in the row, it will appear this way in the final document.

An exception is a column defined in the dictionary with an output format. If a column definition includes an output format, this specification will be applied to the data when it is merged into the script.

You can include formatting specifications in the column flag itself if your script requires it. For example, if you wish to print a column using a specific length or justification, you can include these specifications in the column flag itself. This capability is used to create formatted blocks such as tables, to limit or expand the size of a column in the script, or to override output formats defined in the dictionary.

Note If you use the reserved column flags MRG.DATE and MRG.TIME, you ı specify a format with these flags in order to format the data correctly.

When you want to include a column flag in a script, pressing the [Alt-F3] key di a popup of all available columns. After you select a column from the popup, yoı see another popup of column format specifications. If you wish to designate a sp format for a column, choose one of the options from this popup. You can also di the popup of column format specifications by pressing [Alt-F4].

The format specification must be included in parentheses following the column ı but still within the braces. Some of the column format specifiers require an addit parameter. The parameter provides extra information required by that column fo specification.

For example, the following column flag includes a format specification:

```
{MRG.DATE(OFV,D2/)}
```

In this example, the column format specification OFV is applied to the column fl MRG.DATE. The parameter "D2/" is used to indicate that the current date is to b output in the format MM/DD/YY.

The format specifications available are:

Format Specifier	Meaning
OFV	Specifies a justification, length or conversion, and a limit to the number of multivalues displayed.
NAMECAP	Converts text to first letter uppercase, other letters lowercase.
MV2COMMA	Converts multivalued columns into comma delimited lists.
COMP	Does comparisons of column values against fixed values.
TOUPPER	Converts text to all uppercase.
TOLOWER	Converts text to all lowercase.

Several of the column format specifications require an additional parameter. For example, the OFV specification requires a parameter that indicates the proper justification and length, conversion, or value limit for the column. These are explained below.

OFV

The parameter for this format specifier consists of three parts. The parts are separated with ; (semicolon). They indicate an output format, a justification and length (format), and a limit on the number of values displayed.

For example, the following OFV format specifier might be used in a script:

```
{INVOICE_AMT(OFV,MD2$;R#10;5)}
```

In the example, all three parts of the OFV parameter are included. The parameter MD2,$ specifies that the invoice amount is to have two decimal places and a floating comma and dollar sign. The second part of the parameter indicates that the invoice amount will be right justified in a column 10 characters wide. Finally, the last part of the parameter dictates that only the first five values of the multivalued invoice amount column are to be output.

If one part of the parameter is to be skipped, its place is still held by ; (semicolon). For example, the following format specifier skips the first part of the parameter:

```
{INVOICE_AMT(OFV,;R#10;5)}
```

This example is like the previous one, but does not include a format specification for the output conversion.

For more information on the use of output formats, see the topic on data formatting in the chapter "Creating windows with Paint".

MV2COMMA

This format specifier turns a list of multivalues into a list separated by commas. The parameter added to this specifier can be any word such as "and" or "or" that will be used before the last word in the list.

For example, you might have a multivalued CONTACT column containing this data:

```
Mike
   Joyce
   Zachary
```

The following column flag includes the format specifier MV2COMMA and the parameter "and" for the contact name column above:

```
{CONTACT(MV2COMMA,and)}
```

Using this example, the resulting output would be:

```
Mike, Joyce and Zachary
```

COMP

This format specifier is used primarily when creating conditional blocks. COMP requires a parameter that includes both the comparison operator (=, <, [, etc.) and a value to compare against the contents of the column. The COMP specifier returns a true or false value based on the comparison. That value is in turn used to determine whether the conditional block is printed.

Note When a column is used as a condition, the COMP specifier returns a value of 1 (true) or 0 (false). Put "?" (question mark) as the first character in the column flag to prevent this value from printing.

The following script fragment illustrates the use of the COMP column format specifier:

```
«IF»Since you are {?AGE(COMP,>59)} {AGE} years old,
   don't forget that you are eligible for our senior
   discount!«ENDIF»
```

In the example, the AGE column is used as a condition. When the contents of the column passes the test, the conditional block will print. A "?" (question mark) is used as the first character in the column flag to designate that the AGE column is used as a condition, but that results of the comparison should not to be printed. This prevents the COMP value (a 1) from printing. To print the actual age of the recipient, the AGE column flag must appear again in the conditional block.

The same comparison operators are used for COMP as are used for Query. For more information, see "Using Query" in the chapter "Using windows for queries".

You can use multiple format specifiers in one column flag. If you do, they are processed in order left to right. The output of the first column format specifier becomes the input for the next one.

For example, assume that you have a multivalued column containing the contact names ISABEL, MAX, and ROXANNE. The following example uses two column format specifiers to format this column:

```
{CONTACT(NAMECAP)(MV2COMMA,and)}
```

The end result is this text:

```
Isabel, Max and Roxanne
```

The column format specifier NAMECAP has converted the names into upper and lowercase names. The specifier MV2COMMA has turned the multiple values in this column (after they have been converted into lower case) into a string with commas and the word "and" between the values.

Testing a script

While you are creating a script, you have the means to test your work. This includes a special Merge script debugging tool, as well as the opportunity to test run your merge.

The Merge debugger

The Make Merge window menu, called with [F10] from any prompt in the window, has three script debugging options. There are separate options for debugging the scripts for the heading, footing, and the Merge script itself.

The script debugger reads through the script looking for errors. Errors include invalid column, format or character flags, unmatched sets of flags (for instance, «IF» with no «ENDIF»), or undefined column format specifiers.

When done, the debugger reports the number of errors it has found. To review the errors, enter the prompt for the script, and press [Alt-F7]. This presents a popup of all errors (in reverse order). If you choose an option from the popup, the cursor goes to the error, and you can correct it. Errors that you have already chosen from the debug popup are flagged with an "X".

Testing the Merge template [Alt-F8]

In addition to using the Merge Debugger, you can test your script by actually merging data with it. Two methods are available to do this.

The first is to press [Alt-F8] while editing the Merge script. This keystroke merges one row with the current script and sends the output to the table entered at the *Output* prompt. The resulting document is displayed for your review.

If you did not enter a table name at the *Output* prompt, the merge output is sent to the VOC table using the key of the current row.

When you press [Alt-F8] you are prompted to enter the key of a row to merge with the current template. Enter the key of any row in the data table. Merge reads the row, merges it with the template, writes the document to the output table, and displays it on the screen.

If you have already done an [Alt-F8] review while editing this Merge template, you will see a prompt asking if you wish to review the existing test row. This enables you to see again the row you created earlier. If you enter N at this prompt, a new review document is created and displayed.

When you are done reviewing the sample row, press [Esc] to return to the Merge Script window.

A second method of testing the script is to do a test run. This is an option available from the Make Merge menu. Press [F10] at any prompt to display the menu, and choose the **Testrun** option.

You will see the Merge Test window. There are two prompts: *Option* and *Number of Records*. The *Option* prompt allows you to specify that the output of Merge is to go to the screen (terminal) or to a table, rather than to the printer. Press [F2] or click <Options> to see a list of options available at this prompt. These are the same options as those discussed under "Running a Merge report" below.

The *Number of Records* prompt is used to specify how many rows from the data table are to be used in the Testrun. Merge will randomly pick this number of rows and use them with the Merge template. When the test run is finished, you will be returned to the Make Merge window.

Running a Merge report

Once a Merge template has been created and tested, you are ready to run a Merge report. The process consists of running a command that reads the Merge template, and then reads a row from the data table associated with the template. Data from the row is merged with the template, and the resulting document is sent to the screen, printer, or to an output table. The process of reading and merging the row with the template is repeated until the list of rows to be merged is finished.

Formatting text and resolving flags

In addition to simply outputting the script text, Merge attends to several other tasks. The first of these is to format the paragraphs. Formatting consists of inserting any data required by the column flags, and then applying the paragraph formatting to the resulting text. If the insertion of data with a column flag has resulted in a long paragraph, for instance, Merge reformats the paragraph into as many lines as necessary, and even splits it across pages if required.

Column and format flags are processed right away. Character flags are handled a little differently. These are discussed below under "Merge output".

Merge output

The default behavior of Merge send output to the currently active printer.

You have options, however, to route the Merge output to either the screen (terminal) or to a table. If you send output to the screen, the document created for the printer is displayed on the screen. Paragraph and character formats are interpreted for the currently active printer. Unless you use the **X** (strip) option, this may result in output that does not display properly on the screen.

Output from Merge can also be sent to an Advanced Revelation table. If you choose the **F** option, the formatted text of a document is sent to the table designated at the

Output prompt in the Make Merge window. Merge creates one row in the output table for each row it merges. The key for the merge output is the same as that of the row being merged with the template.

Unlike those sent to the printer or screen, documents in the output table are not ready to print. All page and paragraph formatting is complete, but character formatting such as underlining, bold, and italic lettering remains encoded. Page breaks remain encoded as well. For example, a character flag such as «ITAL» remains in this format when written to the output table. A page break appears as the «FF» flag.

To interpret the character flags, call Merge again with a special option (the **O** option). This option is used to convert these character formats into their respective printer codes while the document is printing or displaying.

Because character flags are not interpreted when output is written to a table, you can review the output without interference from control characters that may appear in the text. In addition, this permits you to create output that is printer independent. The output table can be moved to another computer, and Merge on that copy of Advanced Revelation can interpret the character formats according to the printer available there.

When you call Merge and use the **O** option to process rows in an output table, you only need to specify the name of a template to process. Merge will use only one prompt from the template: *Output*. The *Output* prompt should specify the Merge output table you want to process. The other information—the script, default format, etc.—is already encoded in the output table. You may wish to create a special Merge template called OUTPUT to provide the name of the current output table.

Finally, a special option (the **X** option) creates output that is formatted, but from which all character flags have been stripped. This is useful for reviewing formatted text without seeing the character flags, or for printing to devices that accept no character formats (such as the screen).

Note If you want to create an operating system file ready to be sent to the printer (with all character formatting interpreted), use the TCL PDISK command *before* you begin Merge. Then use Merge with the P option to print to a printer (output will go to the PDISK file). For more information, see PDISK in the chapter "TCL Command reference" in the *Reference* book.

Creating Merge output

Merge is available by selecting the **Run-Report-Merge** options from the Development menu. This displays the Merge window.

At the *Merge Name* prompt enter the name of a Merge template you created earlier using the Make Merge window. Press [F2] or click <Options> to see a list of available Merge templates.

The next prompt allows you to specify options for Merge. The options are used to

override the output specifications in the Merge template. In addition, three of the Merge options enable you to manipulate the character formats and page breaks assigned in the Merge script.

The options acceptable at this prompt are:

Option	Meaning
P	Send Merge output to the printer. Character flags are interpreted as printer codes.
T	Send Merge output to the screen (terminal). Character flags are interpreted as printer codes.
F	Send Merge output to the output table assigned in the Merge template. Character flags and page breaks are encoded but not interpreted.
O	Send output table text to the output device, turning character flags and page breaks into printer codes.
X	Send Merge output to the output device, stripping character flags and interpreting page breaks.

For a discussion of the differences between the table output options, see "Merge output" above.

You can specify more than one option. For example, you might specify both the **T** and **X** options to display a document on the screen (terminal) while stripping character flags.

Related topic

TCL MERGE command

Special features of Merge

A variety of special features can help you customize Merge to your own requirements. These include creating an additional script that can be inserted into the basic script and importing scripts from outside of Advanced Revelation.

Special text

Using the «SPECIAL» flag, you can have Merge insert a script from another table into the current script. This enables you to create custom paragraphs, PS messages,

etc., that can be merged into an otherwise generic script.

When Merge encounters the «SPECIAL» flag, it looks into the table designated at the *Special Text* prompt in the Make Merge window. If a row is found in that table with the same key as the key of the current row, the script from the Special table is inserted at the flag. If there is no row in the Special table with the same key as that of the current row, no insertion is done.

The script in the Special table follows exactly the same conventions as those for ordinary scripts. All formatting from the current script applies to the Special script.

To create the Special text, use the Advanced Revelation full-screen Editor. You will need to insert character, format and column flags by hand, since no popups are available in the Editor to provide these. If your Special text is complex, you can create it using the Merge Script window, and then use the Editor cut and paste buffers to move it to the Special text table.

Default Merge template

You can create a default template using the Make Merge window. When you subsequently create new Merge templates, the values you have specified in the default template will be the default values for the new template.

To create a default template, enter the name DEFAULT at the *Merge Name* prompt. Enter values at all prompts for which you wish to establish defaults, including the scripts. Save your default template by pressing [F9] or click <Save>.

Importing scripts

You are not limited to creating scripts using the *Merge script* prompt in the Make Merge window. You can use any editor that can write out an ASCII text file.

The **Import** option from the Make Merge menu allows you to read an ASCII text file and import it into a Merge template as a script. To import a text file, press [F10] at any prompt to invoke the Make Merge menu.

Choose the **Import** option. When you do, you will be prompted to enter the name of an operating system file. Enter the name, including the full path name.

You will then be asked if you wish to wrap long lines. If you accept the default Y at this prompt, the Merge import process will format long lines to fit into the Merge Script window (80 columns wide). If you enter N at this prompt, long lines will run off the edge of the screen (but will be formatted properly when you merge the document).

Quick Reference Chart for Merge

Keystrokes	Meaning
[F2]	(at a script prompt) Enter script edit window
[F2]	(in a script edit window) Display popup of flag popups
[Alt-F1]	Format flag popup
[Alt-F2]	Character flag popup
[Alt-F3]	Column flag popup
[Alt-F4]	Column format specifiers
[Alt-F5]	Edit column format specifiers popup
[Alt-F7]	Script debug popup
[Alt-F8]	Perform merge sample
[Alt-F9]	Begin format/character flag («)
[Alt-F10]	End format/character flag (»)
[F10]	(at a Make Merge prompt) Display Make Merge menu
{*columnname*}	Insert data from the column *columnname*
{MRG.PAGE}	Insert current page number
{MRG.TIMEDATE}	Insert current time and date
{MRG.DATE}	Insert current date (internal format)
{MRG.TIME}	Insert current time (internal format)

Keystrokes	Format Flags	Meaning
[Alt-F]	«FORMAT»	Establish default format for paragraph
[Alt-U]	«UNFORMAT»	Disable default format for paragraph
[Alt-I]	«IF»	Begin conditional text block
[Alt-L]	«ELSE»	Begin alternate text block
[Alt-E]	«ENDIF»	End conditional or alternate block
[Alt-P]	«PAGE»	Force new page
[Alt-S]	«SPECIAL»	Insert special text
[Alt-V]	«VBLOCK»	Begin multivalued block
[Alt-K]	«END.BLOCK»	End multivalued block
[Alt-H]	«VBLOCK.HEAD»	Define start of value block heading
[Alt-C]	«CENTER»	Center line of unformatted paragraph
[Alt-N]	«ENDCENTER»	End centering of lines
[Alt-R]	«RIGHT»	Right justify remainder of unformatted line
[Alt-J]	«PARA»	Force paragraph without space
[Alt-A]	«PARA.INDENT, n»	Establish paragraph indent of *n* characters
[Alt-W]	«DBLSPC»	Double space lines

Keystrokes	Character Flags	Meaning
Alt-Z,B	«BOLD»	Start bold character
Alt-Z,E,B	«END.BOLD»	End bold character
Alt-Z,U	«ULINE»	Start underline character
Alt-Z,E,U	«END.ULINE»	End underline character
Alt-Z,M	«EMPH»	Start emphasized character
Alt-Z,E,M	«END.EMPH»	End emphasized character
Alt-Z,D	«DBL»	Start double-wide character
Alt-Z,E,D	«END.DBL»	End double-wide character
Alt-Z,S	«SUPER»	Start superscript character
Alt-Z,E,S	«END.SUPER»	End superscript character
Alt-Z,L	«SUB»	Start subscript character
Alt-Z,E,L	«END.SUB»	End subscript character
Alt-Z,I	«ITAL»	Start italic character
Alt-Z,E,I	«END.ITAL»	End italic character
	«FONT1»	Begin font1
	«END.FONT1»	End font1
	«FONT2»	Begin font2
	«END.FONT2»	End font2
	«FONT3»	Begin font3
	«END.FONT3»	End font3

Column Format Specifiers	Meaning
OFV	Specify conversion, justification/length, value limit
NAMECAP	Convert text to first letter uppercase, other letters lowercase
MV2COMMA	Convert multivalued columns to comma-delimited lists
COMP	Compare column values against fixed values
TOUPPER	Convert text to all upper case
TOLOWER	Convert text to all lower case

20. Using Query-By-Example

Introducing Query-By-Example

Query-By-Example (QBE) is Advanced Revelation's premier relational query builder. It lets you quickly and easily search the data in one or more database tables. QBE is useful to novice users who want to search their databases without having to learn and remember complex commands. It is useful to experienced users as a quick and visual way to create and refine queries.

Each table you choose to search is displayed by QBE as a graphic table: each column in the table displays as a column on the screen, and each row in the table displays as a row on the screen. To create a query, you enter selection criteria in the appropriate columns, and press [F9] or click <Save>. QBE builds a Structured Query Language (SQL) query, processes it, and displays the results in the Report Viewer window. QBE's graphic approach to querying lets you use the power of Advanced Revelation's SQL without having to know SQL.

What Query-By-Example can do

QBE adds to the querying power of SQL by providing a graphical display of your query. It also enhances SQL by letting you load R/LIST commands and reuse QBE commands. In particular, QBE lets you:

- Build your queries using existing R/LIST commands and SQL views.
- Save and reuse the QBE queries you create in a special library table.
- Display one or more of the tables available to you in the same window. QBE can display up to 10 tables with up to 300 columns in each table.
- Select which rows to display as results. You select rows by entering selection criteria that compare the data in a column to a value or to another column.
- Include up to 50 columns in the result display. As you select columns, QBE adds them to a window that shows you what the result display will look like.
- Change the order and headings of the results columns, and specify how the rows you display should be sorted.
- Define new columns that select or combine the data from existing columns using equations and functions.
- Display results on your screen or print them.

When to use Query-By-Example

QBE is a relational database tool because it lets you relate the data in various tables and create results that contain information from any or all of the tables you have

related. QBE lets you create queries easily against any number of tables. If you use QBE in conjunction with Advanced Revelation's bonding functions, you can even create complex relational queries against tables that were not specifically designed to support relational querying.

If you are familiar with SQL, you can use it to perform relational queries. If you are less familiar with SQL or if you prefer a visual representation of your queries, you should use QBE to create your queries.

Although you can print and, to some extent, format the results of a QBE query, the tool is not designed to be a report writer. You can produce lists with QBE, but you cannot include totals, headers, or footers in them. You cannot use QBE to produce forms, labels, or merge reports. If you want any of these sorts of reports, you should use one of Advanced Revelation's report tools.

Query-By-Example and SQL

Structured Query Language (SQL) is Advanced Revelation's relational database management tool. It is a set of database commands that you can use interactively or in compiled programs to work with one or more database tables. QBE is an interface between you and SQL. QBE translates the selection criteria you enter in the QBE windows into valid SQL commands. It lets you use the full querying power of SQL without having to know the actual commands.

Seeing the SQL Queries you create

As you enter the criteria of your search, QBE translates them into an SQL SELECT command.

To see the SQL equivalent of your QBE query, press [F3] after you have selected at least one column for the result display. The SQL command QBE displays is for your information only. You cannot edit it.

SQL functions supported by Query-By-Example

SQL includes commands for creating, modifying, and querying databases. QBE uses only the SQL commands for querying (it lets you construct SELECT commands and FROM and WHERE clauses only).

You cannot use any of the following SQL commands with QBE:

CREATE TABLE, CREATE VIEW, DROP TABLE, DROP VIEW, DECLARE, INSERT, UPDATE, DELETE, FETCH, ALTER TABLE, BEGIN TRANSACTION, COMMIT, ROLLBACK, SET TRANSACTION, OPEN, or CLOSE.

Note To enter the SQL functions that create and maintain databases, you can use the SQL window. Choose **DB Admin-SQL** from the Development menu.

Getting help for Query-By-Example

QBE has a complete on-line help system that uses the same keys as the help system available throughout Advanced Revelation. The following table lists the help keys available in QBE:

Key	Function
[F1]	Display context help on the current menu, window, or popup.
[Ctrl-F1]	Display general help on Advanced Revelation.
[Ctrl-F2]	Display conceptual help about QBE.
[F2]	Display a popup listing the options that are currently available to you.
[F6]	Display a popup describing the softkeys that are available in QBE.
[Ctrl-F9]	Display a popup describing all of the keys that are active in QBE.

Enhancing performance with expanded memory

You may experience out-of-memory errors when attempting to use QBE for complex queries. To avoid this problem, we recommend that you make use of expanded memory software when using this tool. Expanded memory provides additional workspace in RAM for applications.

Query-By-Example basics

You can use QBE at a number of levels. If you simply want to view certain columns in a table, there is very little you have to know about QBE. If you want to restrict the rows that are displayed, display columns from more than one table, or change the way the result of your query is displayed, you need to learn a few special words, symbols and procedures. If you want to create complex queries that select rows from many tables based on many comparisons, or if you want to perform calculations on the columns of your tables, then you will have to learn some more advanced procedures. This section explains the basic function of QBE. The section that follows provides details on how to create a query.

How to use Query-By-Example

Before you begin to use QBE, it is helpful to understand its organization and functions. Once you are familiar with the basics of QBE, there are two steps involved in using QBE: create the proper query, and decide how you want the results displayed.

Creating a query

Before you can enter a query, you must decide which tables you want to search. The group of tables you want to search is called the query set. For information about including tables in the query set, see "Adding tables to the query set" later in this chapter.

After you have decided which tables to search, you must decide what query to use. A query selects rows from the tables in the query set. The simplest queries select all of the rows. More complicated queries select rows based on the values they contain in certain columns. The most complicated queries select rows based on comparisons between columns. For more information about queries, see "Creating Query-By-Example queries" later in this chapter.

Displaying the results

Once you have a query that selects the rows you are interested in, you must decide how you want those rows displayed. In particular, you must decide:

- Which columns to include in the result display.
- What order to display the results columns in, and what headings to give them.
- How to sort the results that are displayed (by which columns and in what order).
- Whether to create any new columns that combine the data in existing columns.

All of the procedures for working with the result display are described in "Working with the Result Display" later in this chapter.

Starting Query-By-Example

To start QBE, choose **Run-Report-Query-By-Example (QBE)** from the Development menu. A Welcome to QBE window displays asking for the name of a table. If you know the name of the table you want to work with, enter it in the window. Otherwise, press [F2] or click <Options> to display the Source Data Options popup. You can use the Source Data Options popup to load one or more tables into QBE. Once QBE has started, you can always return to this menu to load a different query, view, or R/LIST command.

To display the Source Data Options popup once QBE is running, press [Esc] until QBE asks you to confirm that you want to lose the current query. Then, press [Enter] to confirm or press [Esc] to return to the current query. If you confirm, QBE displays the Welcome to QBE window. Press [F2] or click <Options> at the window to display the Source Data Options menu.

Loading tables and queries into Query-By-Example

The first step in making a QBE query is to load the tables you want to search. There are several ways to load tables: you can load the tables included in a saved QBE query, SQL view, or R/LIST command by choosing the appropriate option from the Source Data Options popup; you can load tables one at a time from the Current Tables popup, which is accessible from the Source Data Options popup; you can enter a table name or view name in the Welcome to QBE window that appears when you start or exit QBE; and you can add tables to the query set once you have loaded the first table in QBE.

QBE lets you load up to 10 tables with up to 300 columns per table.

Loading tables

To start QBE with a particular table, you can enter the table name in the window that appears when you start QBE. If you can't remember the table name, press [F2] or click <Options> and choose option 1, Table, from the Source Data Options popup. Then, select the table you want to load from the Current Tables popup and press [Enter].

Adding tables to the query set

To load a table once QBE has started:

1. Press [Shift-F8] to display the Add a Table to the Query window.
2. Enter the name of the table you want to load, or press [F2] or click <Options> and select a table from the Current Tables popup.
3. QBE displays the table at the bottom of the Query Set window.

Removing tables from the query set

To remove a table that you have previously loaded, move the cursor to that table and press [F8]. QBE prompts you to confirm your request, and then removes the table and moves the cursor to the next table in the Query Window. You cannot remove a table if it is the only one in the Query Set window.

Loading a saved query

If you have previously saved a QBE query, you can load the saved query from the REPORTS table. For information about saving queries, see "Saving queries" later in this chapter.

To start QBE with a saved Query, choose option 2, Query, from the Source Data Options popup to display the QBE Queries popup. Select the query you want to load and press [Enter]. QBE loads the appropriate tables, and enters the selection criteria specified by the query into the QBE windows.

To load a saved query after QBE is started, display the Source Data Options popup, and choose option 2, Query.

Loading an SQL view

An SQL view is a collection of columns from one or more SQL tables which have been combined into a new derived table. If you have previously created an SQL view, you can load it into QBE. For information about creating SQL views, see the "Building a database" chapter in the *Reference* book.

To start QBE with an SQL view, enter the view name at the Welcome to QBE window that appears when you start or exit QBE. Or choose option 3, View, from the Source Data Options popup to display the QBE SQL Views popup. Select the view you want to load and press [Enter]. QBE loads the view as a single table. All of the relationships between the columns in the view are preserved in its QBE form.

Exploding and collapsing views

A view is displayed as a single table in QBE. To see the tables and relations that created the view, press [F7]. QBE explodes the view to display its base tables. QBE marks, highlights, and adds text to the columns to show the selection criteria that created the view. To collapse the view back to a single table, press [F7] again.

You cannot explode a view that contains an SQL subquery or an SQL GROUP BY, or HAVING clause. If you press [F7] while such a view is loaded, QBE displays an error message.

Loading an R/LIST command

R/LIST is Advanced Revelation's database reporting language. If you have previously saved an R/LIST command from EasyWriter, you can load it into QBE.

QBE cannot translate the selection criteria or formatting portions of an R/LIST command; it translates only the table and output column specifications of the command.

For information about saving R/LIST commands in the REPORTS table, see the "Using EasyWriter" chapter in this book.

To start QBE with a saved R/LIST command, choose option 4, RList, from the Source Data Options popup to display the QBE Existing R/LIST Queries popup. Select the command you want to load and press [Enter]. QBE loads the appropriate table and selects the columns specified by the command for display. QBE ignores the selection criteria and formatting portions of the R/LIST command.

Using the Query-By-Example sample database

To help you get started using QBE, Advanced Revelation comes with a sample database and a set of sample QBE queries. You can use the sample database and queries to follow along with the examples in this chapter.

The sample database

The sample database is stored on a directory called SAMPLE. The database contains information for an automobile parts order-tracking system. It consists of the following tables:

- CAR_PARTS: A list of automobile parts catalogued by part number and part type number.
- CAR_PART_TYPES: A description of each part type number.
- CAR_SUPPLIERS: A list of the names and addresses of parts suppliers catalogued by supplier number.
- CAR_PARTS_SUPP: A list of the supplier number, price, and supplier part number of the suppliers who sell each part.
- CAR_ORDERS: A list of the part orders that have been made.
- CAR_ORDER_ITEMS: A list of the parts contained in each order.

Each of these tables will be further described as needed to illustrate the functions of QBE.

Note Before you can use the sample database you must attach the tables in the SAMPLE directory. To attach these tables, choose **DB Admin-Table-Attach** from the Development menu. At the *Location* prompt, enter SAMPLE.

Working with Query-By-Example windows

All of the functions of QBE take place in three windows: the Query Set window, the Result Set window, and the Report Viewer window. When QBE starts, it displays the Query Set and Result Set windows showing the table, query, or view you chose. When

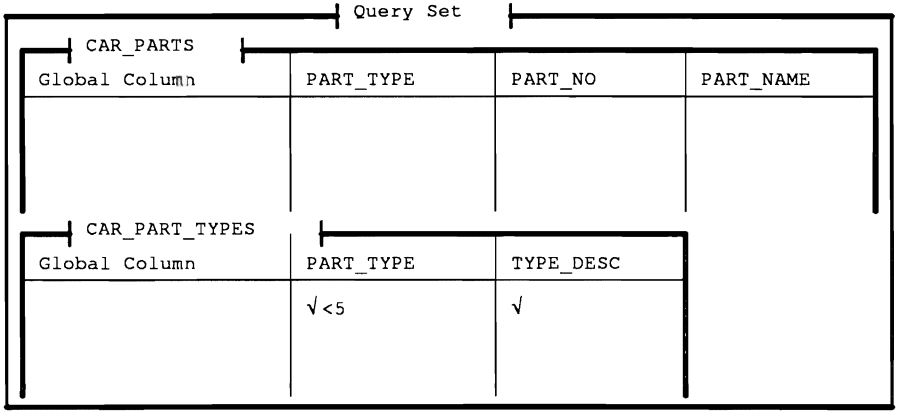
you execute a query, QBE displays the Report Viewer window where you can see the results of your query.

As you create and refine queries, you will need to move within and between these windows to select options and edit text.

The Query Set window

The Query Set window shows the tables you are searching. You use this window to enter the selection criteria of your query, and choose some of the options that determine how the results will be displayed. When QBE starts, it displays the Query Set window at the top of your screen.

QBE displays each table you have loaded in a row/column format, with the table name in the upper left corner. For example, if you have loaded the tables CAR_PARTS and CAR_PART_TYPES, the Query Set window might look like this:



You enter the criteria of your search beneath the headings of the columns they apply to. In the illustration above, for example, the text <5 in the PART_TYPE column of the CAR_PART_TYPES table indicates that only rows where the type number is less than 5 should be selected. The check marks (√) in the PART_TYPE and TYPE_DESC column of the CAR_PART_TYPES table indicate that these columns should be included in the result display.

The first column of each Query set table is always labeled Global Column. You use this column to enter criteria that apply to every column of the table. The columns in the Query Set window are of fixed width.

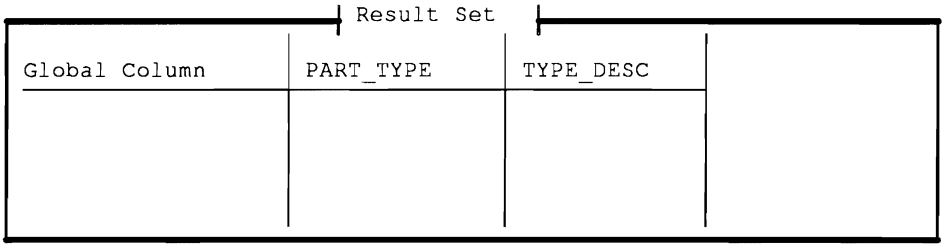
You can load up to 10 tables into the Query Set window. However, because of the size of your screen, QBE cannot display all of them at the same time. In addition, QBE can display only as many columns of the tables as will fit across your screen. For

information about how to scroll to the other columns and tables in the Query Set window, see "Moving between and within windows" later in this chapter.

The Result Set window

The Result Set window shows you how the result display will look. You use this window to see which columns will be included in the result display and to choose most of the options that control how your results will be displayed.

For example, if you have selected the PART_TYPE and TYPE_DESC columns from the CAR_PART_TYPES table to be included in the result display, the Result Set window might look like this:



The columns of the results will be displayed in the order shown in the Result Set table. The columns will also have the headings and sizes they have in this table. For information about changing the headings or order, as well as information about using calculations to add new columns to the results, see "Working with the result display" later in this chapter.

The first column of the Result Set table is always labeled Global Column. You use this column to specify whether to send the results to the printer, and whether to select all or only distinct rows. For information on printing, see "Printing results" later in this chapter. For information about selecting distinct rows, see "Selecting all or distinct rows" also later in this chapter.

The columns in the Result Set window change size to accommodate the display length of the values they contain and their titles. QBE shows only as many columns of the Result Set table as can be displayed across your screen. For information about how to scroll to any other columns, see "Moving between and within windows" later in this chapter.

The Report Viewer window

The Report Viewer window shows the results of your query. QBE displays this window when you execute a query.

The Report Viewer window shows all of the columns you selected in the Query Set window, with the order, headings, and display lengths shown in the Result Set window. For example, suppose the Query Set and Result Set windows look as they do in the preceding figures. When you press [F9] or click <Save> to execute your query, the Report Viewer window would look like this:

```
                        Report Viewer
Page   1                  09:10:11                    07 SEPT 1992
        PART_TYPE          TYPE_DESC
-------------------------------------------------------------------
                     3     Body, fittings
                     4     Ignition system
                     1     Electrical
                     2     Fuel system
```

The first line of the Report Viewer window shows a page number, as well as the time and date the results were displayed. The second line lists the headings of the columns displayed. The last line of the window tells you how many rows met the criteria of your search.

QBE shows only as much of the results as can be displayed on your screen. For information about how to scroll to any portions of the results that are not displayed, see "Moving around in the Report Viewer window" later in this chapter.

The result display is temporary. Like R/LIST reports, the display itself is not saved. To reproduce the display you must execute the query again. You can, however, choose to send the results to the printer rather than to your screen. For more information on printing, see "Printing results" later in this chapter.

To return to the query while the Report Viewer window is displayed, press [Esc]. To redisplay the Report Viewer window, press [F9] or click <Save> to execute the query again.

Moving between and within windows

You use different keys to move around in the Query Set and Result Set windows than you do to move around in the Report Viewer window.

Moving around in the Query and Result Set windows

In the Query Set and Result Set windows, you enter text and choose options at the location of the blinking cursor. Use the following keys to move the cursor to the table and column where you want to enter a criterion or choose an option (the current table is the table where the cursor is):

Key	Function
[Home]	Move to the Global Column of the current table.
[End]	Move to the last column of the current table.
[Enter]	Move right one column.
[Right]	Move right one column.
[Tab]	Move right one column.
[Left]	Move left one column.
[Shift-Tab]	Move left one column.
[Ctrl-Right]	Move right one window.
[Ctrl-Left]	Move left one window.
[PgUp]	Move up one table.
[PgDn]	Move down one table.
[Ctrl-PgDn]	Move to the Result Set window.
[Ctrl-PgUp]	Move to the most recently active table in the Query Set window.
[Ctrl-G]	Move to the table and column in the Query set you choose.
[Ctrl-F]	Search right for the column whose heading contains the text you enter.
[Ctrl-A]	Repeat the last [Ctrl-F] command.
[Letter]	Search right for the heading that begins with the letter you enter.

Moving around in the Report Viewer window

Use the following keys to see any parts of the result display that are not currently visible in the Report Viewer window:

Key	Function
[Home]	Move to the left-most portion of the result display.
[End]	Move to the right-most portion of the result display.
[Right]	Shift the display right one character.
[Left]	Shift the display left one character.
[PgDn]	Move the display down one page.
[Enter]	Move the display down one page.
[PgUp]	Move the display up one page.

Note For more information about the keys you can use in the Report Viewer window, press [Ctrl-F2] while the cursor is in this window.

Using the Query-By-Example Column list

While the cursor is in the Query Set window, you can press [Ctrl-G] to display the QBE Column list. The QBE Column list shows the table name, column name, and column heading for each column in the Query Set window. The list also shows whether the column is selected to be included in the result display.

Use this list to view a summary of the tables that have been loaded and to move the cursor to any table and column in the Query Set window. To move the cursor to a particular column, select it from the list and press [Enter].

Searching for column headings

To search for a column heading in the Query Set window, press [Ctrl-F] or simply start entering the heading you are searching for.

If you press [Ctrl-F], QBE displays the prompt *Text to find* at the bottom of your screen. Enter the first part or all of the column heading you want to find and press [Enter]. QBE searches for the first occurrence of the character string in a column heading to the right of the cursor in the current table. If QBE finds a match, it moves the cursor to the matching column. The text you enter can be any part of the column heading you are searching for.

If you start entering while the cursor is in the Query Set window and you are not in edit mode, QBE displays the Text to find prompt and enters the characters you enter after it. Press [Enter] to search the column headings to the right of the cursor, or press [Esc] to cancel the search.

To repeat the last search you made, press [Ctrl-A]. QBE executes the last [Ctrl-F] command you entered.

Viewing the dictionary

A table's dictionary contains information about the content and format of each of its columns. To see the Advanced Revelation dictionary entry for a column, move the cursor to the column and press [Shift-F1]. QBE displays the Dictionary Info popup showing the specifications for the column.

This popup is for your information only and cannot be used to change the dictionary entry.

Entering and editing in Query-By-Example

In QBE, you enter and edit text to specify selection criteria and calculations or to change the heading of a column in the Result Set window. With a few exceptions, you use the same keys to work with text in QBE windows as you use in any other part of Advanced Revelation. General instructions for working with text are presented in the Getting Started section in the main documentation.

There are two types of text in QBE: literal text and example text. You use literal text to enter values, comparison operators, formulas, and column headings. You use example text to link columns together and create calculations. Example text is highlighted on your screen. Literal text is not.

You use the [F4] and [Shift-F4] keys to work with literal and example text:

- [F4] starts and ends edit mode. The exact function of this key changes depending upon where the cursor is. If you press [F4] in a blank column in the Query Set window or while the cursor is over literal text, QBE switches to literal text edit mode. If you press [F4] while the cursor is over example text, QBE switches to example text edit mode. If you press [F4] while you are already in edit mode, QBE exits from edit mode.
- In the Query Set window, [Shift-F4] switches between example and literal text edit modes. The exact function of this key changes depending upon where the cursor is. If you press [Shift-F4] while the cursor is in a blank column or over literal text, QBE switches to example text edit mode. If you press [Shift-F4] while you are in example text edit mode or the cursor is over example text, QBE switches to literal text edit mode.
- Regardless of the kind of text you are entering, once you are in edit mode, you can use the following editing keys:

Key	Function
[Right]	Move the cursor right one character.
[Left]	Move the cursor left one character.
[Ctrl-Home]	Move the cursor to the beginning of the line.
[Ctrl-End]	Move the cursor to the end of the line.
[Home]	Move the cursor to left-most character in the column.
[End]	Move the cursor to the right-most character in the column.
[Up]	Move the cursor up one line in the column.
[Down]	Move the cursor down one line in the column.
[Enter]	Accept text, exit Edit mode, and move the cursor right one column.
[Del]	Delete character at current cursor position.

(continued)

Key	Function
[Backspace]	Delete character to the left of the cursor.
[Ins]	Insert/overwrite toggle.
[Ctrl-Y]	Delete the word to the right of the cursor.
[Ctrl-X]	Delete the current line.
[Ctrl-L]	Delete to the end of the line.
[Ctrl-K]	Delete from the beginning of the line to the cursor.

Saving queries

QBE maintains a table called REPORTS, where you can create a library of QBE queries. For information about loading queries you have saved, see "Loading a saved query" earlier in this chapter.

When you are ready to save a query you have created, press [Shift-F9] to display the QBE Save Query window. Enter a name and description for your query and then press [F9] or click <Save>. QBE saves the Query Set and Result Set windows as they are currently set up. It saves the query, any results characteristics you have set and all tables that are currently loaded.

QBE cannot save a query unless at least one column has been created or selected for the result display.

Printing results

Usually, QBE displays the results of your queries on your screen. To print the results instead, press [Shift-F10]. QBE displays the phrase To Printer" in the Global Column of the Result Set window. When you press [F9] or click <Save> to execute a query, QBE will print the results instead of displaying them.

QBE formats the display to fit on your printer's specified page size. Thus, the page divisions that you see when you display results on your screen may not be the same as those you see when you print the results.

To redirect results to your screen, press [Shift-F10] again.

Making simple queries

The complexity of the QBE query you create depends on the information you want to extract from your tables. In many cases, your query will consist of no more than a few simple selection criteria.

The following examples show a progression from the simplest query, which displays an entire table, to a slightly more complex query which displays only some of the columns and rows of the table. These examples use the sample table CAR_SUPPLIERS. To use this table, you must attach the tables in the SAMPLE directory. For more information, see "The sample database" earlier in this chapter.

Displaying an entire table

To display an entire table, you load it, select all of its columns for the result display, and press [F9] or click <Save>. For example, the following query displays the entire CAR_SUPPLIERS table:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	ADDRESS	CITY
	√	√	√	√

To create this query:

1. Choose Run-Report-Query-By-Example (QBE) from the Development menu.

2. At the Welcome to QBE window, enter CAR_SUPPLIERS in the window and press [Enter].

   QBE loads the table, displays it in the QBE Query Set window, and displays the cursor in the Global column.

2. To select all columns of the table for the result display, press [Ins] while the cursor is in the Global column.

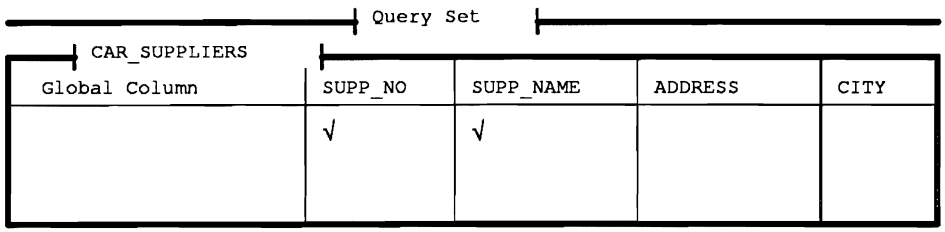
   QBE enters a check mark in each column of the table and adds the columns to the table in the Result Set window.

3. To execute the query, press [F9] or click <Save>.

4. QBE shows you the steps it is performing at the bottom of your screen. When the query has been successfully evaluated, QBE displays the results in the Report Viewer window.

Displaying only some columns

In many cases, you may want to see only some of the columns in a table. For example, the following query displays only the SUPP_NO and SUPP_NAME columns of the CAR_SUPPLIERS table:



To create this query:

1. Load the CAR_SUPPLIERS table into QBE as described in the previous example.

   If the table is already in QBE, you do not need to load it again, but you do need to deselect any selected columns by moving the cursor to the Global column and pressing [Del].

2. Move the cursor to the SUPP_NO column.

   To move there from the Global column, you can press [Right] or [Enter]. You can also move to the column by choosing it from the QBE Column List ([Ctrl-G]) or by searching for its heading ([Ctrl-F]).

3. To select the column for the result display, press [Ins].

   QBE places a check mark in the column and adds it to the table in the Result Set window.

4. Move the cursor to the SUPP_NAME column and press [Ins] to add this column to the result display.

   QBE places a check mark in the column and adds it to the end of the table in the Result Set window.

5. To execute the query, press [F9] or click <Save>.

   QBE shows you the steps it is performing at the bottom of your screen. When the query has been successfully evaluated, QBE displays the results in the Report Viewer window.

All of the keys for moving around in the Query Set and Result Set windows are listed when you press [Ctrl-F9]. They are also described in Moving Between and Within Windows" earlier in this chapter.

Displaying only some rows

Usually, you want to see the information in only some of the rows in a table. To select certain rows to display, you enter search criteria. For example, the following query uses the search criteria to display only the rows of the CAR_SUPPLIERS table that have a SUPP_NO value that is less than five:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	ADDRESS	CITY
	<5	√	√	√

To create this query:

1. Load the CAR_SUPPLIERS table into QBE as described in the previous example.

   If the table is already in QBE, you do not need to load it again, but you do need to deselect any selected columns by moving the cursor to the Global column and pressing [Del].

2. To select all columns of the table for the result display, press [Ins] while the cursor is in the Global column.

   QBE enters a check mark in each column of the table and adds the columns to the table in the Result Set window.

3. Move the cursor to the SUPP_NO column.

   To move there from the Global column, you can press [Right] or [Enter]. You can also move to the column by choosing it from the QBE Column List ([Ctrl-G]) or by searching for its heading ([Ctrl-F]).

4. To select rows with a Supplier number less than 5, press [F4] to switch to edit mode, then enter <5.

   This search criterion tells QBE to select rows where SUPP_NO is less than 5.

5. To execute the query, press [F9] or click <Save>.

   QBE shows you the steps it is performing at the bottom of your screen. When the query has been successfully evaluated, QBE displays the results in the Report Viewer window.

QBE has many types of selection criteria you can use to specify precisely the rows you want to see. The criteria you can enter are described in the next section.

Creating Query-By-Example queries

A QBE query is a set of selection criteria that select rows from one or more database tables. QBE offers four basic types of selection criteria:

- You can select rows by comparing one or more of their columns to one or more values. For example, you can select rows whose date column contains a date from last year.
- You can select rows by comparing one or more of their columns to other columns. This is one of the most important functions of QBE because it allows you to combine information from two or more related tables. For example, if one table contains general information about an order and another contains specific information, you can join the two tables and display general and specific information about an order.
- You can select rows by comparing them to the value in a particular row. For example, you can select rows that have a purchase date later than the row that contains the name Becky Schwartz in its customer name column.
- You can select the specific row that contains the minimum or maximum value in a particular column. For more information about this type of selection criterion, see "Finding the minimum or maximum value" later in this chapter.

Before you create any type of query, you must load the tables you want to search into QBE. For more information, see "Loading tables and queries into Query-By-Example" earlier in this chapter.

Query guidelines

The following sections describe the general rules you should follow when you enter a query.

Case sensitivity

QBE is not case sensitive to operators (such as OR, LIKE, NOT, BETWEEN, etc.) and functions (such as MIN, SUM and COUNT). You may enter these words in uppercase or lowercase letters. In this manual, they are shown in uppercase for emphasis.

QBE is case sensitive to the actual words and phrases you are searching for. Therefore, be sure to capitalize the words and phrases you search for in exactly the same way they appear in the table.

Quotation marks

If your selection criteria include any multi-word phrases, enclose the complete phrase in quotation marks (<169>). For example, to select columns that contain the phrase Fuel system, you would enter:

```
"Fuel system"
```

Reserved words

If your selection criteria are the same as any of QBE's reserved words, enclose the criteria in quotation marks ("). For example, to select columns that contain the abbreviation for Oregon, OR, which is reserved, you would enter:

```
"OR"
```

The following words are reserved whether you enter them in uppercase or lowercase letters: AVG, COUNT, IN, IS, LIKE, MAX, MIN, NULL, SUM, BETWEEN, AND, OR, NOT, DISTINCT, and UNIQUE.

If you are unsure whether something is reserved, or you receive an error when you execute the query, enclose the criteria in quotation marks.

Special data formats

QBE is sensitive to the format of the data you enter as selection criteria.

For example, when you enter selection criteria in date columns you may be able to use a variety of formats depending upon how the column was originally defined. To be sure that you are using a correct format, use the same format as you would use to add a new date into the table.

Selecting columns to display

In order to execute a query, you must specify at least one column for the result display (unless the query contains a calculation that creates a column). On the other hand, if you have selected at least one column for the result display, you can execute a query with no selection criteria. Such a query displays all rows and the selected columns of the table. For example, to display all of the information in the PART_NAME column of the CAR_PARTS table, you can simply select the column and press [F9] or click <Save>.

Adding columns to the result display

To select a column, move the cursor to the column and press [Ins]. QBE enters a check mark (√) in the column and adds it to the end of the result set table.

To select all of the columns in a table, move the cursor to the table's Global column and press [Ins]. QBE enters check marks (√) in all of the table's columns and adds them to the Result Set table. You can include up to 50 columns from as many as 10 tables to be displayed as results.

Removing columns from the result display

To remove a column from the result display, move the cursor to the column and press [Del].

To remove all of the columns in a table that are currently selected, move the cursor to the table's Global column and press [Del].

Comparing columns to values

The most common database queries select rows that have a certain value or range of values in a specific column. For example, you might want to select all type 1 parts (PART_TYPE = 1) or all orders greater than $100 (PRICE > 100).

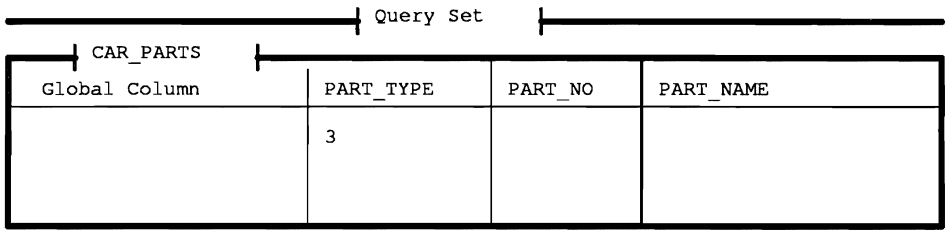
To tell QBE to compare a column in the Query Set window column with a value, move to the column, and then enter a comparison operator followed by the value. To specify that the column should be equal to the value, you need not enter a comparison operator.

The following table describes the comparison operators you can use in a query:

Operator	Function
= or none	Equals
<	Is less than
>	Is greater than
<=	Is less than or equal to
>=	Is greater than or equal to
<>	Is not equal to

Selecting rows with a particular value

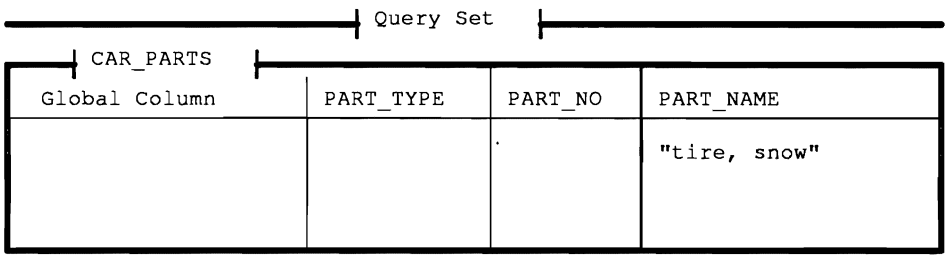
To select rows that have a particular value in a certain column, you simply enter the value in that column. For example, the following query selects rows that have a part type value of 3:


Query Set

CAR_PARTS

Global Column	PART_TYPE	PART_NO	PART_NAME
	3		

The following query selects rows in the CAR_PARTS table that have the phrase tire, snow" in the PART_NAME column:


Query Set

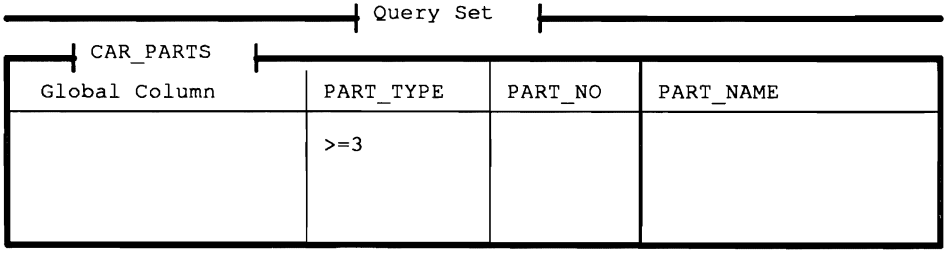
CAR_PARTS

Global Column	PART_TYPE	PART_NO	PART_NAME
			"tire, snow"

The phrase is enclosed in quotation marks ("") because it contains a space. If you are searching for a single unreserved word, quotation marks are not necessary. For more information about reserved words, see "Reserved words" earlier in this chapter.

Selecting rows greater or less than a value

You can use the QBE comparison operators in columns that contain numbers, words, or dates. For example, the following query selects parts with a type greater than or equal to 3 from the CAR_PARTS sample table:


Query Set

CAR_PARTS

Global Column	PART_TYPE	PART_NO	PART_NAME
	>=3		

This query selects the suppliers in the CAR_SUPPLIERS sample database whose names begin with a letter greater than M:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	ADDRESS	CITY
		>M		

This query selects orders from the CAR_ORDERS sample table with an order date before March 15, 1992:

Query Set

CAR_ORDERS

Global Column	ORDER_NO	SUPP_NO	DATE_ORD
			<"MAR 15 1992"

The date is enclosed in quotation marks ("") because it contains spaces. It is enterd in the same format as you would use to enter new dates into the CAR_ORDERS table.

For more information about entering dates, see "Special data formats" earlier in the chapter.

Selecting rows similar to a value

If you know part of the value you are searching for in a column, you can use the following phrase to tell QBE to find rows that contain values like the one you specify:

```
LIKE "value"
```

Value is a representation of the value you want to find. It has the following properties:

- It must be surrounded by quotation marks ("").
- Due to case sensitivity, characters must be capitalized in exactly the same way they appear in the table.

- It can have the percent sign (%) in any position as a wild card which stands for any number of characters.
- It can have the underscore character (_) in any position as a wild card which stands for a single character.

For example, to find the suppliers in the CAR_SUPPLIERS table whose names contain the word Import, you could enter the following query:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	ADDRESS
		LIKE "%Import%"	

Because there is a percent sign before and after Import, QBE finds rows with these words at any position in the SUPP_NAME column.

To find rows that contain the word Import or Export, you could enter the following query:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	ADDRESS
		LIKE "% __port%"	

The percent signs indicate that any text may precede or follow the word, and the underscore (_) indicates that QBE should consider any character in that position a match (Import, import, export, Export, etc.).

If the item you are searching for contains the literal percent (%) or underscore (_) characters, you must precede those characters with a backslash (\). This indicates that these are literal characters rather than wild cards.

For example, to search for the name Turbo_Carb, you would enter:

```
LIKE "Turbo\_Carb"
```

To search for rows that contain the phrase %max, you would enter:

```
LIKE "%\%max%"
```

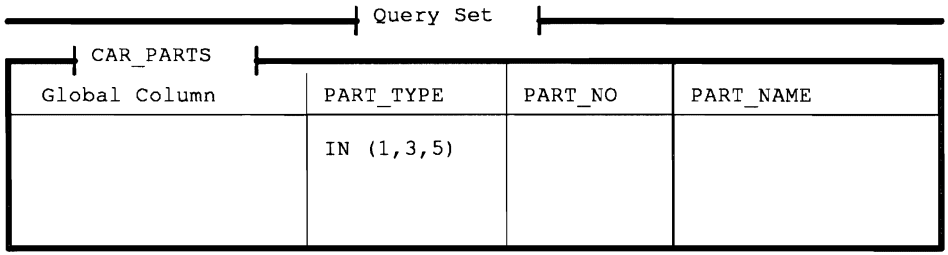
Selecting rows from a set of values

If you want to select rows that contain any one of a set of values you specify, enter the following phrase in the column of interest:

```
IN (value1, value2, value3, ...)
```

Value1, value2, and value3 are the values you want to search for. Each value must follow the rules described in Query Guidelines" earlier in this chapter.

For example, to select rows from the CAR_PARTS table that have a type number of either 1, 3, or 5, you could enter the following query:



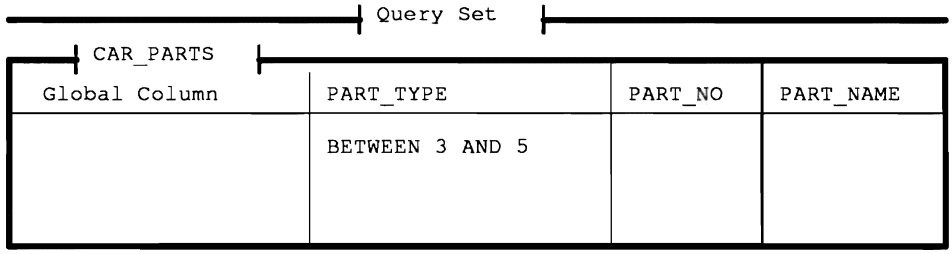
You can produce the same effect using the OR operator. For more information, see "Making multiple comparisons in the same column" later in this chapter.

Selecting rows between two values

To select rows that have values in a certain range, enter the following phrase in the column that contains the values:

```
BETWEEN value1 AND value2
```

Value1 and value2 define the range of values you want to search for. For example, to search for part types between 3 and 5, in the CAR_PARTS table, you could enter the following query:



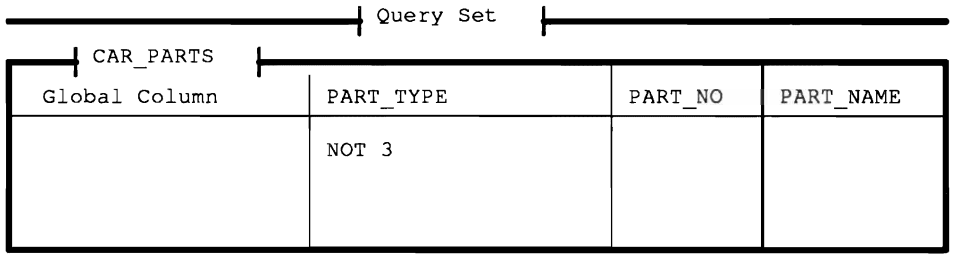
QBE selects the rows with PART_TYPE values greater than or equal to 3 and less than or equal to 5.

Excluding rows with a value

To select only those rows that do not contain a certain value in a column, enter the following phrase in the column with the value you want to exclude:

```
NOT value
```

Value is the value you want to exclude. For example, to select all rows except those with a part type of 3, you could enter this query:



You can also use NOT to change the meaning of other QBE functions. For example, you can enter NOT LIKE, NOT BETWEEN, or NOT NULL.

Selecting rows with no value

To select rows that have no data in a certain column, enter the following phrase in the column:

```
NULL
```

For example, to find the suppliers in the CAR_SUPPLIERS table for whom you have no address, you could enter the following query:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	ADDRESS	CITY
			NULL	

Making multiple comparisons in the same column

Usually, you can enter a single selection criterion in a column to find the values you want there. Sometimes, however, you may need two or more criteria to select the proper rows. For example, you may want to select columns that are either empty or contain a value of zero. QBE lets you enter any number of criteria in a single column.

Before you enter a second criterion in a column, you must decide whether the column must meet both conditions (criterion one AND criterion two) or either condition (criterion one OR criterion two).

Criterion one and criterion two

To indicate that a column must meet two or more selection criteria, separate the criteria with a comma (,) character.

For example, to specify in the CAR_PARTS table that type number must be greater than or equal to 2 and less than or equal to 5, you could enter the following query:

Query Set

CAR_PARTS

Global Column	PART_TYPE	PART_NO	PART_NAME
	>=2, <=5		

This query has the same effect as the BETWEEN operator discussed earlier. To search the CAR_SUPPLIERS table for suppliers that have the words Import and Motors somewhere in their names, you could use the following query:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME
		LIKE "%Import%",LIKE "%Motors%"

The phrase LIKE %Import%" selects rows with the word Import somewhere in their name. Likewise, LIKE %Motors%" selects rows with the word Motors somewhere in their name. Because the two criteria are separated by a comma, the name must contain Import and Motors somewhere in the name. In contrast to the query LIKE %Import Inc.%" described earlier, in this query the two words may be next to each other or widely separated in the name and the word Motors may appear before or after Import.

Criterion one or criterion two

To indicate that a column can meet one or the other criteria, separate the criteria with the word OR. For example to find parts with a type number of 1, 3 or 5, you could enter the following query:

Query Set

CAR_PARTS

Global Column	PART_TYPE	PART_NO	PART_NAME
	1 or 3 or 5		

The above query has the same effect as the IN operator discussed earlier. To search for suppliers in Paris or Seattle, you could enter this query:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	CITY
			Paris OR Seattle

Making multiple comparisons in different columns

So far, all of the query examples have had selection criteria in only one column. You can also enter criteria in more than one column that compares each column to a separate value. For information about comparing columns to each other, see "Comparing columns in a table" later in this chapter.

When you enter a criterion in a second column, you are imposing both conditions (criterion one AND criterion two).

Note You cannot impose an either condition (criterion one OR criterion two) by entering criterion in a second column.

To indicate that a row must meet the criteria in two or more columns, enter criteria in different columns. For example, to find parts in the CAR_PARTS table that have a type number of 1 and a part number between 1 and 5, you could enter the following query:

Query Set

CAR_PARTS

Global Column	PART_TYPE	PART_NO	PART_NAME
	1	BETWEEN 1 AND 5	

To find suppliers in the CAR_SUPPLIERS table who are in Seattle and have the word Import in their names, you could enter the following query:

Query Set

CAR_SUPPLIERS

Global Column	SUPP_NO	SUPP_NAME	CITY
		LIKE "%Import%"	Seattle

Relating tables with a join

You join two or more tables to make use of the relation between the tables, or, in other words, to create a relational query. You join two tables by a common column. For example, you can relate the CAR_PARTS and CAR_PART_TYPES tables by joining them by the PART_TYPE column that they both have.

Methods for joining columns

There are two different but equivalent ways to join columns: you can choose options from popups that guide you through the joining process, or you can mark the columns you want to compare directly by typing the same example text in both.

Example text allows you to more quickly see which columns you have joined. On the other hand, the join popups give you a cleaner display where you can more easily see the selection criteria you have entered. You can use either or both of these methods in any query; the choice is yours.

Regardless of the method you use or the number of tables you are working with, there are two basic steps to create a join:

1. Select the columns you want to compare.
2. Select the kind of comparison you want to make (column one equals column two, column one less than column two, etc.).

Using the Query-By-Example join popups

QBE has a set of popups that you can use to guide you through the process of joining columns. When you press [Ctrl-F6], QBE presents you with a popup that lists your options for creating or removing joins in the current column. When you have marked a column for a join using the popups, QBE highlights its heading.

To view a list of the columns you have joined and the comparisons you are making, press [Ctrl-F6] and choose option 3, Review. QBE displays the Defined Multi-table

Joins popup which shows the columns you have joined and which operator you specified.

The general method for joining columns with the join popups is as follows:

1. Mark the first column for the join: move the cursor to it, press [Ctrl-F6], and choose option 1, Mark, from the popup that QBE displays.
2. Mark the second column for the join: move the cursor to it, press [Ctrl-F6], and choose option 2, Link, to join the column to the other marked column.

   QBE displays the Complete Link popup, which lists all of the columns that have been marked for a join.
3. Choose the column you want to join the current column to from the Complete Link popup.

   QBE displays the QBE Join Operator popup showing the various comparisons you can make between the two columns you have marked.
4. Choose the comparison you want to make from the Join Operator popup.

Note Specific examples of how to use the join popups to compare columns are presented below.

Using example text

If you prefer a more visual approach to joining columns, you can use example text. Each piece of example text is a single word which marks columns in the Query Set window that you want to compare. QBE creates a join between any two columns that contain the same word of example text.

Example text looks just like the literal text you enter to specify selection criteria except that example text appears highlighted on your screen. The procedures for editing both example and literal text are described in "Entering and editing in Query-By-Example" earlier in this chapter.

The general procedure for joining columns with example text is:

1. Move the cursor to the first column and press [Shift-F4] to enter example text edit mode.

   QBE changes the cursor and displays the word EXAMPLE on the Status Line.
2. Enter any single word that will help you remember the purpose of the join.

   You can use exactly the same keys to edit example text as you can to edit literal text.
3. Move the cursor to the second column of the join, press [Shift-F4], and enter the same word preceded by any of the comparison operators listed in Comparing columns to values" earlier in this chapter.

Note Specific examples of how to compare columns using example text are presented below.

A two-table join

When you compare columns in two or more tables, you create a relational query. It is relational because the result of the query depends upon the relation between the tables you compare.

The most useful multi-table joins select rows when the joined columns in two or more different tables contain the same value. You can use this type of join to combine the information in two tables that are related by a common column.

For example, CAR_ORDERS and CAR_ORDER_ITEMS are directly related. For each automobile parts order, there is one row in the CAR_ORDERS table, and a set of rows in the CAR_ORDER_ITEMS table. The single row in the CAR_ORDERS table has information about the whole order. The set of rows in the CAR_ORDER_ITEMS table have detailed information about each item that was ordered.

The two tables are linked together by the ORDER_NO (order number) column. Every order in the CAR_ORDERS table has a unique order number. There can be multiple rows in the CAR_ORDER_ITEMS table with the same order number. For example, order No. 1 has a single row in the CAR_ORDERS table that lists the order number, supplier number, date, total price, and other information about the order. In the CAR_ORDER_ITEMS table, order No. 1 has three rows. Each row lists the order number, part number, quantity ordered and other information about a single part in the order.

You can use the relation between these two tables to combine their information into a single result display that shows general and specific information about an order.

If you use the QBE popups, you can look at all of the information in order No. 2 using this query:

Query Set

CAR_ORDERS			
Global Column	ORDER_NO	SUPP_NO	DATE_ORD
	√2	√	√
CAR_ORDER_ITEMS			
Global Column	ORDER_NO	LINE_ITEM	SUPP_PART_NO
		√	√

If you use example text, a query to examine order No. 2 will look like this:

Query Set

CAR_ORDERS			
Global Column	ORDER_NO	SUPP_NO	DATE_ORD
	√ getorder ,2	√	√
CAR_ORDER_ITEMS			
Global Column	ORDER_NO	LINE_ITEM	SUPP_PART_NO
	getorder	√	√

Note The following sections describe how to make the joins shown in the previous two queries.

Using the popups to make the join

For this example, you need to join the ORDER_NO columns in the CAR_ORDERS and CAR_ORDER_ITEMS tables. To use the join popups to make this join:

1. Load the CAR_ORDER and CAR_ORDER_ITEMS tables into QBE.

Note Use [Shift-F8] to add tables to the Query Set in QBE.

2. Move the cursor to the ORDER_NO column in the CAR_ORDERS table and press [Ctrl-F6] to mark this column for a join.
3. Choose option 1, Mark, from the popup.

   QBE highlights the column title to show that it is part of a join.
4. Press [PgDn] to move the cursor to the ORDER_NO column in the CAR_ORDER_ITEMS table. Press [Ctrl-F6], then choose option 2, Link, to complete the join.
5. Choose the ORDER_NO column in the CAR_ORDERS table from the Complete Link popup.
6. Choose option 1, Equals, from the Join Operators popup to specify that you want to select rows that have the same order number.

You can press [Ctrl-F6] and choose the option Review to display a popup showing the kind of join you created.

Using example text to make the join

For this example, you need to join the ORDER_NO columns in the CAR_ORDERS and CAR_ORDER_ITEMS tables. To use example text to make this join:

1. Move the cursor to the ORDER_NO column of the CAR_ORDERS table, press [Shift-F4] to switch to example text edit mode, and enter `getorder`.

   The exact word you enter is not important as long as it exactly matches the word you enter in the other column you want to join this one to. The word you enter symbolizes the join you are making.
2. Move the cursor to the ORDER_NO column of the CAR_ORDER_ITEMS table, press [Shift-F4] to switch to example text edit mode, and enter `getorder`.

Entering selection criteria

Regardless of the technique you used to join CAR_ORDERS and CAR_ORDER_ITEMS, you can use this procedure to select the rows that apply to order No. 2:

1. Move the cursor to the ORDER_NO column in either table, and press [Shift-F4] to switch to edit mode.

Note If you used the popups to make the join, you will need to press [Shift-F4] again to switch from example text edit mode to literal text edit mode.

   It doesn't matter which table you enter the selection criterion in, QBE will apply it to both tables.

2. While in literal text edit mode, enter the number 2, to tell QBE to select the rows for order number 2.

   If you used example text to make the join, enter a comma (,) after the word `getorder` and then enter the number 2. The comma specifies that you want a logical AND between the two conditions.

Displaying the results

Once you have created the join and entered the criteria of your search, you need only select the appropriate columns for the result display.

To select columns for the result display and execute the query:

1. Move the cursor to the Global column of the CAR_ORDERS table and press [Ins] to include all its columns in the result display.

   QBE lists the columns in the table in the Result Set window.

2. Move the cursor to the Global column of the CAR_ORDER_ITEMS table and press [Ins] to include its columns in the result display.

   QBE adds the columns in this list to the table in the Result Set window.

3. Move the cursor to the ORDER_NUMBER column in the CAR_ORDER_ITEMS table and press [Del] to remove it from the result display.

   This step ensures that the order number appears only once in the result display.

4. Press [F9] or click <Save> to execute the query.

   At the bottom of your screen, QBE shows the steps it is taking. When the query process is complete, QBE displays the Report Viewer window with your result display.

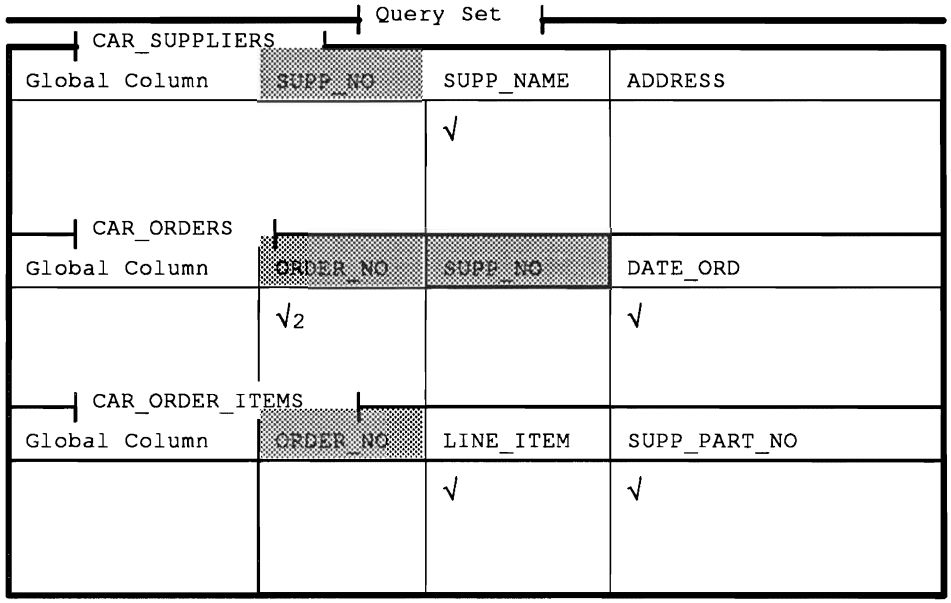
A three-table join

You use the same basic procedure whether you are joining two, three, or any number of columns in any number of tables.

In the previous example, the CAR_ORDERS and CAR_ORDER_ITEMS tables were joined through their ORDER_NO columns to produce a result display that combined related information from both tables. To make an even more informative result display, you can add information from a third table.

Because the CAR_SUPPLIERS and CAR_ORDERS sample tables both have a SUPP_NO column, you can join the two tables by this column and then substitute the supplier's name for the supplier's number in the result display.

If you use the QBE popups, your query will look like this:



Using the popups to make the joins

For this example, you need to join the ORDER_NO columns in the CAR_ORDERS and CAR_ORDER_ITEMS tables, and to join the SUPP_NO columns in the CAR_ORDERS and CAR_SUPPLIERS tables. To use the join popups to make these joins:

1. Load the CAR_ORDERS, CAR_ORDER_ITEMS, and CAR_SUPPLIERS tables into QBE.

   Note Use [Shift-F8] to add tables to the Query Set in QBE.

   Note that you can only see two tables at a time. For information on moving between tables, see "Moving between and within windows" earlier in this chapter.

2. Follow the procedure in the previous example to create a join between the ORDER_NO columns in the CAR_ORDERS and CAR_ORDER_ITEMS tables.

3. Move the cursor to the SUPP_NO column in the CAR_ORDERS table and press [Ctrl-F6] to mark this column for a join.

4. Choose option 1, Mark, from the popup.

   QBE highlights the column title to show that is part of a join.

5. Move the cursor to the SUPP_NO column in the CAR_SUPPLIERS table, press [Ctrl-F6], then choose option 2, Link, to complete the join.

6. Choose the SUPP_NO column in the CAR_ORDERS table from the Complete Link popup.
7. Choose option 1, Equals, from the Join Operators popup to specify that you want to select rows that have the same supplier number.

You can press [Ctrl-F6] and choose the option Review to display a popup showing the kind of joins you created.

Entering selection criteria

For this example you need to select the rows for order No. 2. To enter the proper selection criterion:

1. Move the cursor to the ORDER_NO column in either the CAR_ORDERS or CAR_ORDER_ITEMS table, and then press [F4] to switch to edit mode.

   It doesn't matter which table you enter the selection criterion in, QBE will apply it to both tables.
2. Enter the number 2, to tell QBE to select the rows for order No. 2.

Displaying the results

Once you have created the joins and entered the criteria of your search you need only select the appropriate columns for the result display.

To select columns for the result display and execute the query:

1. Move the cursor to the SUPP_NAME column of the CAR_SUPPLIERS table and press [Ins] to include this column in the result display.

   This step makes the supplier name the first column in the result display. No other columns from this table will be listed.
2. Move the cursor to the Global column of the CAR_ORDERS table and press [Ins] to include all its columns in the result display.

   QBE lists the columns in the Result Set window.
3. Move the cursor to the SUPP_NO column in the CAR_ORDERS table and press [Del] to remove this column from the result display.

   The supplier number need not be shown in the results because the supplier name is already shown.
4. Move the cursor to the Global column of the CAR_ORDER_ITEMS table and press [Ins] to include its columns in the result display.

   QBE adds the columns in this list to the table in the Result Set window.
5. Move the cursor to the ORDER_NO column in the CAR_ORDER_ITEMS table and press [Del] to remove it from the result display.

This step ensures that the order number appears only once in the result display.

6. Press [F9] or click <Save> to execute the query.

   At the bottom of your screen, QBE shows the steps it is taking. When the query process is complete, QBE displays the Report Viewer window with your result display.

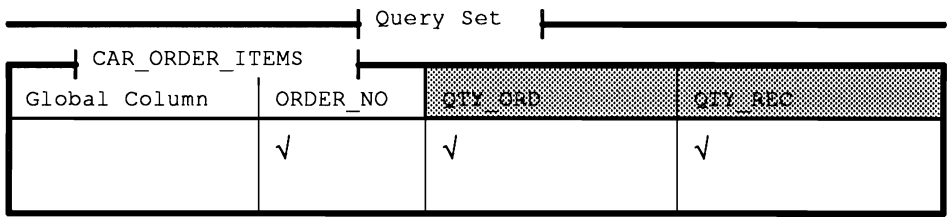
Comparing columns in a table

In addition to selecting rows by comparing columns to values, you can also select rows by comparing columns to each other. The procedure by which you create a comparison is just like the procedure you use to join two tables.

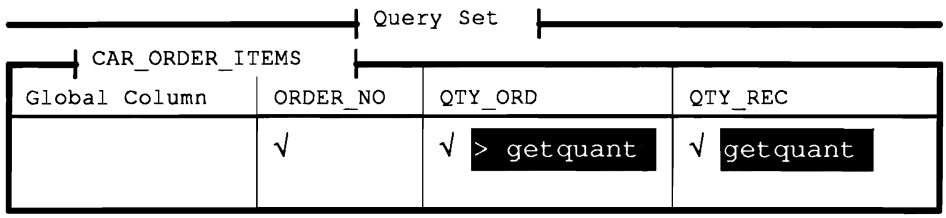
When you compare two columns in the same table, you tell QBE to select rows where one column is greater than, less than, or equal to the other. For example, you can select rows in the CAR_ORDER_ITEMS table where the number of parts ordered is greater than the number received (QTY_ORD QTY_REC).

You can use either of the two joining methods to compare columns. For example, the CAR_ORDER_ITEMS sample table provides details about each order that has been made to a supplier. To find out if any of the parts you ordered were not yet received, you can select all rows where QTY_ORD (quantity ordered) is greater than QTY_REC (quantity received).

If you use the QBE popups, your query will look like this:



If you use example text, your query will look like this:



Note The following sections describe how to make the comparisons used in the previous two queries.

Using the popups to make the comparison

For this example, you need to compare the QTY_ORD and QTY_REC columns of the CAR_ORDER_ITEMS table. To use the Join popups to make this comparison:

1. Load the CAR_ORDER_ITEMS table, then move the cursor to the QTY_REC column and press [Ctrl-F6] to mark this column for a comparison.
2. Choose option 1, Mark, from the popup.

   QBE highlights the column title to show that is part of a comparison.
3. Move the cursor to the QTY_ORD column, press [Ctrl-F6], then choose option 2, Link, to complete the comparison.

   QBE displays the Complete Link popup.
4. Choose QTY_REC from the Complete Link popup.

   QBE displays the Join Operator popup.
5. Choose option 3, Greater, to specify that you want to select rows where QTY_ORD is greater than QTY_REC.

   QBE completes the comparison and highlights the column title to show that it is part of a comparison.

You can press [Ctrl-F6] and choose the option Review to display a popup showing the type of comparison you created between the two columns.

Using example text to make the comparison

For this example, you need to compare the QTY_ORD and QTY_REC columns of the CAR_ORDER_ITEMS table. To use example text to make this comparison:

1. Move the cursor to the QTY_REC column, press [Shift-F4] to switch to example text edit mode, enter `getquant`, and press [Enter].

The exact word you enter is not important as long as it exactly matches the word you enter in the other column you want to compare this one to. The word you enter symbolizes the comparison you are making.

2. Move the cursor to the QTY_ORD column, press [Shift-F4] to switch to example text edit mode, and enter `getquant`.
3. Press [Home] to move the cursor to the beginning of the column, press [Shift-F4] to switch to literal text mode and enter the greater than character (>).

   The highlighted text `getquant` tells QBE that the two columns should be compared. The greater than sign (>) in the QTY_ORD column indicates that QBE should select rows where QTY_ORD is greater than QTY_REC.

Displaying the results

Regardless of how you created the comparison, you use the same procedure to finish the query.

To select columns to be included in result display and execute the query:

1. Move the cursor to the Global column of the CAR_ORDER_ITEMS table and press [Ins].

   QBE adds a check mark to each column in the table and adds the columns to the table in the Result Set window.
2. Press [F9] or click <Save> to execute the query.

   At the bottom of your screen, QBE shows the steps it is taking. When the query process is complete, QBE displays the Report Viewer window with your result display.

Working with the result display

QBE lets you change the result display to show only the information you need. You can also change the way information is displayed. In particular, you can:

- Select which columns to include in the result display. For more information about selecting columns, see "Selecting columns to display" earlier in this chapter.
- Change the headings and display order of the results columns.
- Sort the result display by a primary and one or more secondary sort columns.
- Create new calculated columns that combine the data in existing columns.

Sorting the results

QBE lets you sort the results of a query in ascending or descending order. You can specify a primary and any number of secondary sort columns. QBE uses the primary sort column to order the result display. If two or more rows have the same primary value, QBE orders them based on the values in the first secondary sort column. If the secondary values are the same, QBE uses the next secondary sort column, and so on.

To mark a column for a sort, move the cursor to the column and press [Shift-F2] for an ascending sort, or [Shift-F3] for a descending sort. To remove the column from the sort, press [Shift-F2] or [Shift-F3] again. When you choose a column for a sort, QBE automatically includes it in the result display. QBE considers the first column you select as the primary sort column. This column is marked by a 1" in the Result Set window. The next column you select is the first secondary sort column and is marked by a 2" in the Result Set window. The next column you select is marked 3", and so on. You can change the order of the sort columns by pressing [Ctrl-F5].

Note QBE orders values based on the active language set, which is specified in the International Environment window. To display this window, choose **Options-Configuration-Environment-International** from the Development menu.

Sorting on a single column

The following query displays an alphabetized list of the rows in the CAR_PARTS sample table:

Query Set

CAR_PARTS

Global Column	PART_TYPE	PART_NO	PART_NAME
	√	√	↑√

Result Set

Global Column	PART_NAME	PART_TYPE	PART_NO
	1 ↑		

To create this query:

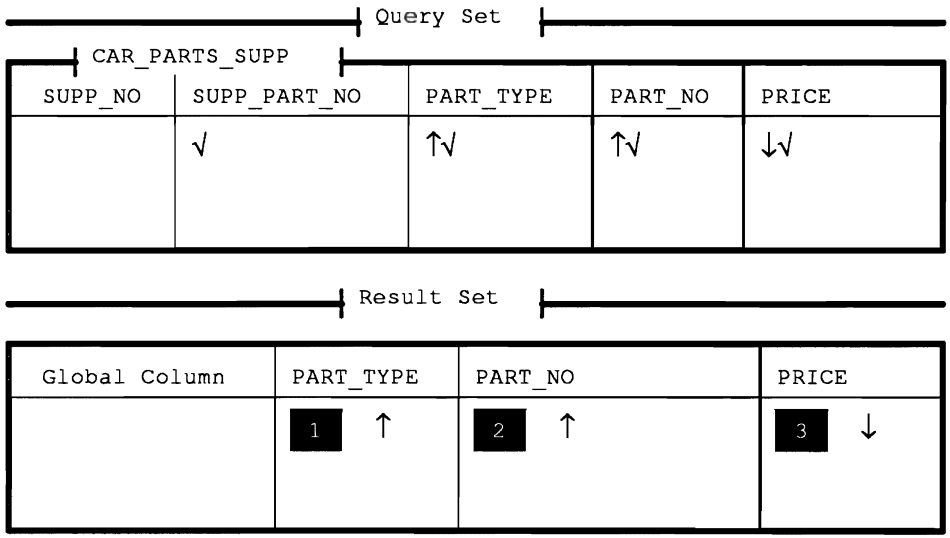
1. Load the CAR_PARTS table into QBE.
2. Move the cursor to the PART_NAME column and press [Shift-F2] to mark the column for an ascending sort.

   QBE adds an up-arrow character (↑) to the column and includes the column in the result display. QBE marks the PART_NAME column in the Result Set window with a 1" to show that it is the primary sort column.
3. Move the cursor to the Global column of the CAR_PARTS table and press [Ins] to include the remaining columns in the result display.

   QBE lists these columns after the PART_TYPE column in the Result Set window.
4. Press [F9] or click <Save> to execute the query.

Sorting on multiple columns

The following query displays a list of the rows in the CAR_PARTS_SUPP sample table, sorted by increasing part type. Where part types are the same, the list is sorted by increasing part number. Where part type and part numbers are the same, the rows are sorted by decreasing price.



To create this query:

1. Load the CAR_PARTS_SUPP table into QBE.
2. Move the cursor to the PART_TYPE column and press [Shift-F2] to mark the column for an ascending sort.

   QBE adds an up-arrow character to the column and includes the column in the result display. QBE marks the PART_TYPE column in the Result Set window with a 1" to show that it is the primary sort column.
3. Move the cursor to the PART_NO column and press [Shift-F2] to mark the column for an ascending sort.

   QBE adds an up-arrow character to the column and includes the column in the result display. QBE marks the PART_NO column in the Result Set window with a 2" to show that it is the first secondary sort column.
4. Move the cursor to the PRICE column and press [Shift-F3] to mark the column for a descending sort.

   QBE adds a down-arrow character to the column and includes the column in the result display. QBE marks the PRICE column in the Result Set window with a 3" to show that it is the second secondary sort column.
5. Move the cursor to the Global column of the CAR_PARTS_SUPP table and press [Ins] to include the remaining columns in the result display.

   QBE lists the remaining columns of the CAR_PARTS_SUPP table after the three sort columns in the Result Set window.

Changing the precedence of the sort columns

QBE orders the sort columns as you mark them in the Query Set window. One way to create a particular sort precedence is to mark the columns in the order you want them to affect the result display.

Once you have marked a set of columns, you can change their precedence by pressing [Ctrl-F5].

For example, in the previous example the precedence of the sort columns was: PART_TYPE, PART_NO, then PRICE.

To change the primary sort column from PART_TYPE to PRICE:

1. Press [Ctrl-F5] to display the Re-assign Order popup.
2. Move the cursor down to highlight CAR_PARTS_SUPP.PRICE, then press [Enter] to copy the line to the Sort column of the popup.

   This line now appears in both columns of the popup.
3. Move the cursor up until the first line of the popup is highlighted and then press [Enter] to copy the column to the top of the list.

4. Press [F9] or click <Save> to exit from the popup and change the precedence of the sort columns.

   You can press [Esc] to exit from the popup without reordering the columns.

Rearranging the columns of the results

QBE shows the columns in the result display in the order they are shown in the Result Set window. Thus, to change the order of the columns in the result display, you must change their order in the Result Set window.

QBE adds columns to the Result Set window in the order you mark them in the Query Set window. One way to create a particular column order is to mark the columns for inclusion in the result display in the order you want to see them.

Once the Result Set window lists columns, you can reorder them by pressing [Shift-F7]. For example, if you load the CAR_SUPPLIERS table and select all of the columns by pressing [Ins] in the Global column, the table in the Result Set window will list the columns in the same order as they are shown in the Query Set window (with ADDRESS and CITY after SUPP_NO and SUPP_NAME).

To reorder the columns so that the supplier's address and city are listed first:

1. Press [Shift-F7] to display the QBE Results Order popup, which shows the present order of the columns in the Result Set window.
2. Move the cursor down to highlight CITY, then press [Enter] to copy the line to the Sort column of the popup.

   The word CITY now appears in both columns of the popup.
3. Move the cursor up until the first line of the popup is highlighted and then press [Enter] to move the column to the top of the list.

   QBE reorders the list with CITY at the top.
4. Repeat steps 2 and 3 to move the ADDRESS line to the top of the list.

   QBE reorders the list with ADDRESS at the top followed by CITY.
5. Press [F9] or click <Save> to exit from the popup and reorder the columns in the Result Set window.

   You can press [Esc] to exit from the popup without reordering the columns.

Changing the headings or display width of results columns

QBE shows the columns in the result display with the headings they have in the Result Set window. Thus, to change the headings of columns in the result display, you must change their headings in the Result Set window.

To change the heading or display width of a column in the Result Set window:

1. Move the cursor to the column you want to change and press [F4].

   QBE displays the Result Column Heading window which shows the current heading of the column.

2. At the *Column heading* prompt, edit the heading that is displayed, enter a new one, or press [F2] or click <Options> to display selection options.

3. At the *Display width* prompt, you can enter a number to indicate how many characters wide this column should be in the result display.

4. Press [F9] or click <Save> to exit the window and change the heading or width of the column. Or press [Esc] to exit the window without making your changes.

For information about changing the heading of a column you have created yourself, see "Modifying a created column" later in this chapter.

Selecting all or distinct rows

Duplicate rows contain the same values in every column you have selected for the result display. QBE usually includes any duplicate rows it finds in the result display. To tell QBE not to display duplicate rows, press [Shift-F6]. QBE adds the word Distinct" to the Global column in the Result Set window.

To return to the default condition, press [Shift-F6] again to remove the word Distinct" from the Global column.

Including calculated columns in the results

QBE gives you a set of mathematical operators and functions that let you make calculations on the values in one or more columns of a table. The results of each calculation are listed in a new column in the result display. You can create and name this column yourself by entering the calculation in the Result Set window. Or you can have QBE create and name the new column by entering the calculation in the Query Set window.

Creating calculated columns in the result set window

When you create a column from the Result Set window, QBE prompts you for a column heading and a calculation. The calculation includes mathematical operators and functions, and the headings of columns you want to make calculations on.

To create a calculated column from the Result Set window:

1. Move the cursor to the column in the Result Set window after which you want to place the new column.
2. Press [Ins] to display the QBE Calculated Column window.
3. Enter a heading for the column at the *Column heading* prompt and press [Enter].
4. At the *Display width* prompt, press [Enter] to accept the default or enter a number to indicate how many characters wide this column should be in the result display.
5. Enter the calculation you want to make at the *Calculation* prompt and press [F9] or click <Save>.

   QBE adds the column you created to the table in the Result Set window.

Note Examples of calculations you can enter in the Result Set window appear later in this section.

Entering calculations

After naming a column, you enter the calculation you want to perform.

You can enter any part of the calculation in uppercase or lowercase letters.

If a column you include in the calculation has the same heading as another column from a currently loaded table, QBE will prompt you to specify which table you mean.

You can also specify the table name with the column name in the calculation. For example, because both the CAR_PARTS and CAR_PART_TYPES tables have a column named PART_TYPE, to refer to the PART_TYPE column in the CAR_PARTS table, you could enter:

```
CAR_PARTS.PART_TYPE
```

Modifying a created column

To change the column heading or calculation in a column you have created, move the cursor to that column and press [F4]. QBE redisplays the Calculated Column window for this column. When you are finished editing, press [F9] or click <Save> to save your changes.

QBE displays only as much of the calculation as will fit in the column. To see the entire calculation, you can move to the column, and press [F4] to switch to edit mode. Then press [F3] while the cursor is at the *Calculation* prompt to display the entire calculation at the top of your screen.

Deleting a created column

To delete a column you have previously created, move the cursor to the column and press [Del]. QBE removes the column from the table.

Creating calculated columns in the Query Set window

When you enter a calculation in any column of the Query Set window, QBE creates a new column at the end of the result display. The calculation includes mathematical operators and functions, and example text that specifies which columns the calculation works with. Columns you create in the Query Set window are shown in the result display with no heading. If you want the column to have a name, you must create it from the Result Set window.

To make a calculation from the Query Set window:

1. Mark each column you want to use in the calculation with a word of example text.

   Use a different word for each column. Each word is like a variable where data from its column will be stored.

2. Enter your calculation in any column. Wherever you want to include a column in the calculation, refer to it with the same word of example text that you used to mark that column. For more information, see "Entering calculations" below.

   Separate the calculation from any other text in the column with a comma(,) character.

Note Examples of calculations you can enter in the Query Set window appear later in this section.

Entering calculations

When you enter a calculation in the Query Set window, the functions and operators must be in literal text, while the text that identifies the columns in the calculation must be in example text.

Editing and deleting calculations

You use the same keys to edit or delete calculations in the Query Set window as you use to enter any other text. The only difference is that when you enter a calculation, you must switch between literal and example text. For more information about

switching between example text edit mode and literal text edit mode, see "Entering and editing in Query-By-Example" earlier in this chapter.

QBE displays only as much of the calculation as will fit in the column. To see the entire calculation, you can move to the column, switch to edit mode and press the arrow keys to scroll right and left in the calculation.

Using operators

You can use any of the following mathematical operators to make calculations on the values in a table:

Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division
||	Concatenation

Using the mathematical operators

You can use one or more of QBE's mathematical operators in any query to extend the information in your result display.

For example, to see the effect of a 20 percent import tax on the parts prices of your suppliers, you could create a column that adds 20 percent to each price in the CAR_PARTS_SUPP table. You can create such a column from the Result Set or Query Set window.

From the Result Set window

To create a new column that adds 20 percent to the prices in the CAR_PARTS_SUPP table, you could move the cursor to the Result Set window, press [Ins], and fill in the prompts at the Calculated Column window as shown below:

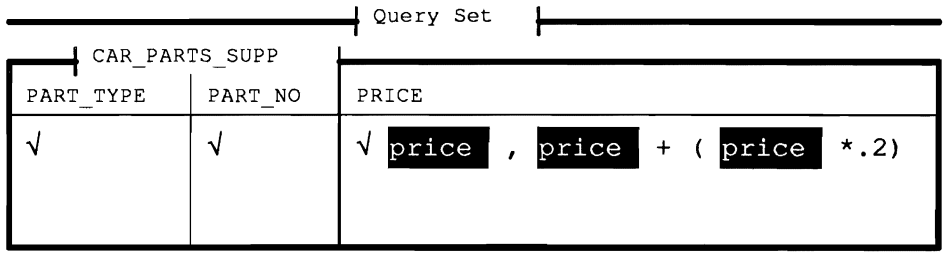
Prompt	Entry
Column heading	Price with Tax
Display width	14
Calculation	PRICE + (PRICE * .2)

If there were another column named PRICE loaded into QBE, the calculation would have to show the table name as well as the column heading for this column. When necessary, QBE prompts you for the table name.

Press [F9] or click <Save> to create the column and insert it following the column with the cursor.

From the Query Set window

To create a new column that adds 20 percent to the prices in the CAR_PARTS_SUPP table, you could enter the following query in the Query Set window:



To create this query:

1. Mark the PRICE column with the example text `price`.

   For information about typing example text, see "Entering and editing in Query-By-Example" earlier in this chapter.

2. Enter a comma (,) followed by the following expression:

   ```
   price + (price * .2)
   ```

Both instances of the word `price` must be highlighted as example text, and the remaining characters must be literal text. QBE automatically switches to literal text edit mode when you press a space or enter a non-alphabetic character. However, to switch back to example text edit mode, you must press [Shift-F4].

3. Move the cursor to the Global column and press [Ins] to include all the columns of the table in the result display.

Using the concatenation operator

You use QBE's concatenation operator to put together the values in two or more columns. For example, you can create a column that lists the address, city and country of the suppliers together. You can create such a column from the Result Set or Query Set window. Note that the values combined to create the column are concatenated together with no space between values.

Be aware that when you concatenate columns using the Query Set window, the results are not only concatenated together but are truncated where the information is too long. If you use the Result Set window you can specify the display width and can insert literal characters (,") between the values from different columns.

From the Result Set window

To create a new column that concatenates the values in the ADDRESS, CITY, and COUNTRY columns of the CAR_SUPPLIERS table, you could move the cursor to the Result Set window, press [Ins], and fill in the prompts at the Calculated Column window as shown below:

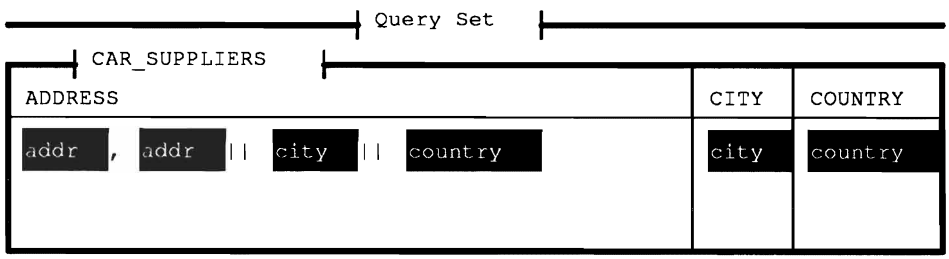
Prompt	Entry
Column heading	Entire Address
Display width	40
Calculation	ADDRESS || CITY || COUNTRY

Press [F9] or click <Save> to create the column and insert it following the column with the cursor.

Note that you could also concatenate literal characters enclosed in quotation marks to separate the values in each column (for example, address||","||city||","||country).

From the Query Set window

To create a new column that concatenates the values in the ADDRESS, CITY, and COUNTRY columns of the CAR_SUPPLIERS table, you could enter the following query in the Query Set window:



To create this query:

1. Mark the ADDRESS, CITY, and COUNTRY columns with the example text `addr`, `city`, and `country`, respectively ([Shift-F4]).

For information about typing example text, see "Entering and editing in Query-By-Example" earlier in this chapter.

2. Move the cursor to the ADDRESS column, press [F4] to enter example text edit mode, and then press [Right] to move the cursor past the word `addr`.

   You can also enter the calculation in any other column of the table.

3. Enter a comma (,) followed by the following expression:

   ```
   addr || city || country
   ```

The words `addr`, `city`, and `country` must be example text, the remaining characters must be literal text. You need to press [Shift-F4] again after you enter the || characters before typing the example text.

4. Press [Ins] while the cursor is in the SUPP_NO, SUPP_NAME, and PHONE columns to include these columns in the result display.

5. Press [F9] or click <Save> to execute the query.

   You need not select the ADDRESS, CITY and COUNTRY columns because they will be displayed together in the new column.

Note .The information from each column is concatenated together without any spaces and is truncated at the end. If you use the Result Set window to concatenate columns you can widen the display and specify literal characters between columns.

Using functions

QBE's functions make a calculation on the rows you select in the column where you enter the function. You can use any of the following functions to make calculations on the values in a table:

Function	Purpose
MIN	Displays the minimum of a group of values.
MAX	Displays the maximum of a group of values.
SUM	Displays the total of a group of values.
AVG	Displays the average of a group of values.
COUNT	Displays a count of the number of values in a group of rows.

In this manual the functions are shown in uppercase for emphasis. When you enter them, you can enter them in uppercase or lowercase.

You can use functions in two ways:

- If you use a function without selecting any columns to be included in the result display, QBE calculates the function once for all of the values in the rows you selected. The sections below describe this way of using functions.
- If you use a function and select columns to be included in the result display, QBE groups the rows according to the values in the non-function columns you select. It then calculates the function once for each group. For more information about grouping values, see "Grouping values" later in this chapter.

To use a function:

1. Enter the selection criteria that select the rows you are interested in.
2. Enter the function you want to use in the column you are interested in.
3. If you want to group values, move the cursor to the column you want to group by, and press [Ins] to include it in the result display.

Finding the minimum or maximum value

The MIN and MAX functions find the minimum or maximum value in the column and rows you select. You can use these functions in columns that contain numbers or text.

For example, you could use the MIN function to select the row in the CAR_PARTS_SUPP table that contains the carburetor with the lowest price.

From the Query Set window

To find the lowest priced carburetor you can enter the following query:

Query Set

CAR_PARTS_SUPP

SUPP_NO	SUPP_PART_NO	PART_TYPE	PART_NO	PRICE
		2	10	MIN

The 2 in the PART_TYPE column and the 10 in the PART_NO column select the rows that concern carburetors. The word MIN in the PRICE column displays the lowest price in the selected rows of this column.

Because no columns are selected for the result display, QBE displays a single minimum for the values in the rows you selected.

From the Result Set window

Rather than typing the word MIN in the PRICE column, you can enter the selection criteria for a carburetor (PART_TYPE of 2 and PART_NO of 10) in the Query Set window and then move the cursor to the Result Set window to create a minimum column with a title.

While the cursor is in the Result Set window, you can press [Ins] and fill in the prompts at the Calculated Column window as shown below:

Prompt	Entry
Column heading	Minimum Price
Display width	13
Calculation	MIN(PRICE)

Press [F9] or click <Save> to create the column and insert it following the column with the cursor.

Summing Values

The SUM function calculates the total of the values in a particular column of the rows you select. You can use this function only in columns that contain numbers. For example, you can use the SUM function to find out how much you have paid a particular supplier.

From the Query Set window

The following query uses the TOTAL_PRICE column of the CAR_ORDERS table to display the total price of all orders you have made from supplier number 5:

Query Set

CAR_ORDERS

Global Column	ORDER_NO	SUPP_NO	TOTAL_PRICE
		5	SUM

The 5 in the SUPP_NO column selects all rows that concern supplier number 5, and the word SUM in the TOTAL_PRICE column tells QBE to display a total of the selected rows in this column.

Because no columns are selected for the result display, QBE displays a single sum for the values in the rows you selected.

From the Result Set window

Rather than typing the word SUM in the PRICE column, you can enter the selection criterion for a supplier number 5 in the Query Set window and then move the cursor to the Result Set window to create a SUM column with a title.

While the cursor is in the Result Set window, you can press [Ins] and fill in the prompts at the Calculated Column window as shown below:

Prompt	Entry
Column heading	Orders from Supplier # 5
Display width	24
Calculation	SUM(TOTAL_PRICE)

Press [F9] or click <Save> to create the column and insert it following the column with the cursor.

Averaging values

The AVG function displays the average of the values in the column and rows you select. You can use this function only in columns that contain numbers. For example, you can use the AVG function to find out the average cost of orders you made before March 1, 1992 from the CAR_ORDERS table.

From the Query Set window

The following query displays the average price of all orders in the CAR_ORDERS table made before March 1. 1992:

Query Set

CAR_ORDERS

Global Column	ORDER_NO	DATE_ORD	TOTAL_PRICE
		<"mar 1 1992"	AVG

The expression <mar 1 1992" in the DATE_ORD column selects all rows that have a date before March 1. 1992. The word AVG in the TOTAL_PRICE column tells QBE to display an average of the selected rows in this column.

From the Result Set window

Rather than typing the word AVG in the PRICE column, you can enter the selection criterion for dates before March 1, 1992 in the Query Set window and then move the cursor to the Result Set window to create an average column with a title.

While the cursor is in the Result Set window, you can press [Ins] and fill in the prompts at the Calculated Column window as shown below:

Prompt	Entry
Column heading	Average Order before 3/1/91
Display width	27
Calculation	AVG(TOTAL_PRICE)

Press [F9] or click <Save> to create the column and insert it following the column with the cursor.

Counting Rows

The COUNT function displays the total number of values in the column and rows you select. In general, the function counts only the rows with some value in the column you select.

However, if you include an asterisk (*) with the function, it will also include rows in which the column contains no value. You can use this function in columns that contain numbers or words.

For example, you could use the COUNT (*) function to find out how many different suppliers sell carburetors.

From the Query Set window

The following query displays a count of the suppliers in the CAR_PARTS_SUPP table who sell carburetors:

Query Set

CAR_PARTS_SUPP

SUPP_NO	SUPP_PART_NO	PART_TYPE	PART_NO	PRICE
COUNT (*)		2	10	

The 2" in the PART_TYPE column and the 10" in the PART_NO column select the rows that concern carburetors. The phrase COUNT (*)" in the SUPP_NO column tells

QBE to display the number of rows selected. If you used the word COUNT" QBE would not include a selected row if it had no value in the SUPP_NO column.

From the Result Set window

Rather than typing the phrase COUNT (*) in the PRICE column, you can enter the selection criteria for carburetors in the Query Set window (2 and 10) and then move the cursor to the Result Set window to create a total column with a title.

While the cursor is in the Result Set window, you can press [Ins] and fill in the prompts at the Calculated Column window as shown below:

Prompt	Entry
Column heading	Carburetor Suppliers
Display width	20
Calculation	COUNT(SUPP_NO)

Press [F9] or click <Save> to create the column and insert it following the column with the cursor.

Combining functions

For clarity, the preceding examples show only one function per query. However, you can easily include two or more functions in a single query or combine a function with another mathematical operator.

Combining two or more functions

You can include as many functions in a query as you like. For example, to find the minimum, maximum and average price of a carburetor, you can enter the phrase MIN, MAX, AVG in the PRICE column of the CAR_PARTS_SUPP table, or create three columns in the Result Set window for the three functions. You can also combine functions as needed using various operators.

Grouping values

If you enter a function and, in addition, select a column to be included in the result display, QBE will group the rows you select according to the values in the column you select. For example, if you select the PART_TYPE column, QBE will group the data in the table according to the part type numbers. All type 1 rows will be grouped together, all type 2 rows will be grouped together, and so on.

For each different group, QBE calculates the function you enter. The result display shows an ascending list of the groups QBE created and the value of the function for each group.

Grouping on a single column

Often it is sufficient to group the data in a table based on a single column. For example, to find the largest order you have made to each supplier, you can enter the following query in the CAR_ORDERS table:

Query Set

CAR_ORDERS

Global Column	ORDER_NO	SUPP_NO	DATE_ORD	TOTAL_PRICE
		√		MAX

The check mark in the SUPP_NO column tells QBE to group data based on supplier numbers and to display one line of results for each different supplier number it finds. The word MAX in the TOTAL_PRICE column displays the maximum value in this column for each of the groups QBE creates.

Rather than typing MAX in the TOTAL_PRICE column, you can create a maximum column from the Result Set window.

Grouping on multiple columns

You can group by as many different columns as you like. QBE treats group columns exactly as it does sort columns. The first column you select is the primary group column. The next one you select is the first secondary group column, and so on. If two rows have the same value in the primary column, QBE creates subgroups based on the values in the first secondary column.

For example, to display a list of the lowest price for each part in the database, you could enter the following query in the CAR_PARTS_SUPP table:

Query Set

CAR_PARTS_SUPP

SUPP_NO	SUPP_PART_NO	PART_TYPE	PART_NO	PRICE
		√	√	MIN

The check marks in the PART_TYPE and PART_NO columns tell QBE to group data based on each unique part type/part number class and to display one line of results for

each different combination. This results in one line of results for each part in the database because each part has a different part number/part type combination.

The word MIN in the PRICE column displays the lowest value in this column for each of the groups that QBE creates. Rather than typing MIN in the PRICE column, you could create a minimum column in the Result Set window.

To create a result display that is easy to read, you should make the PART_TYPE column the primary group column. To do so, select the PART_TYPE column first, or rearrange the results columns following the procedure described in "Rearranging the Columns of the Results" earlier in this chapter.

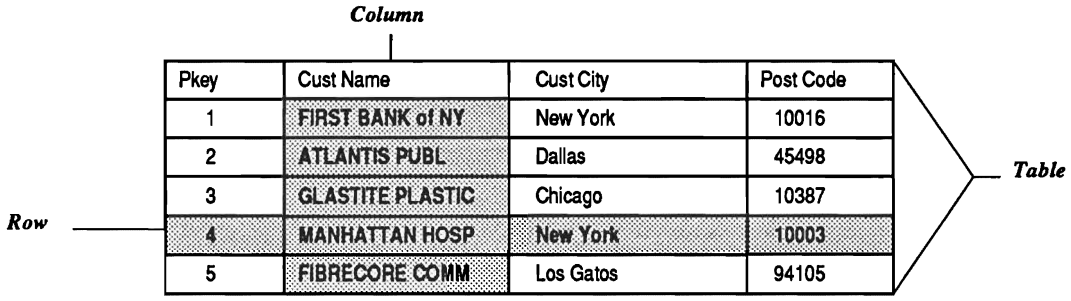
Managing the database

21. About databases in Advanced Revelation

Introduction to tables, rows, and columns

In Advanced Revelation, you store data in tables. A table has individual entries called rows. For example, each customer in a customer table is thought of as being in a row of that table.

Rows are in turn divided into columns that contain individual elements of information for the rows:



Pkey	Cust Name	Cust City	Post Code
1	FIRST BANK of NY	New York	10016
2	ATLANTIS PUBL	Dallas	45498
3	GLASTITE PLASTIC	Chicago	10387
4	MANHATTAN HOSP	New York	10003
5	FIBRECORE COMM	Los Gatos	94105

Data is represented as being in tables that are divided into columns and rows.

Variable-length rows and columns

In contrast to traditional tables, tables in Advanced Revelation can contain rows and columns that vary in size from entry to entry:

1	FIRST BANK of NEW YORK	New York
2	ATLANTIS PUBLICATIONS	Dallas
3	GLASTITE PLASTIC	Chicago
4	MANHATTAN HOSPITAL	New York
5	FIBRECORE COMMUNICATIONS	Los Gatos

In Advanced Revelation, rows and columns don't have to be the same length.

This gives Advanced Revelation great flexibility in storing data. For example, it is easy to store text data or even programs in an Advanced Revelation table:

19 Feb 92	Dear Sirs,	I recently saw your ad for computer software, and

FOR CTR = 1 TO 10	PRINT CTR * 5	NEXT CTR	CALL MSG("Done!")

Text or other variable data is stored with each entry (letter, document, program, etc.) as a row in the table, and each line as a "column".

There are two major advantages to having variable-length rows and columns:

- You save space because you don't need to guess in advance how long the longest piece of data will be for each column and pad the shorter columns with blanks.
- Tables are very flexible, because you can add or delete columns or change their characteristics at any time, even after data has been entered in the table. Changes do not require you to alter structure of the table in any way.

Size limits

The overall limit is that no row can exceed 65,536 bytes in length. By extension, no column can be longer than that either. Each empty column in a row takes up one byte, so you can have one column containing 65,535 characters, or 65,535 columns containing no data, or anything in between.

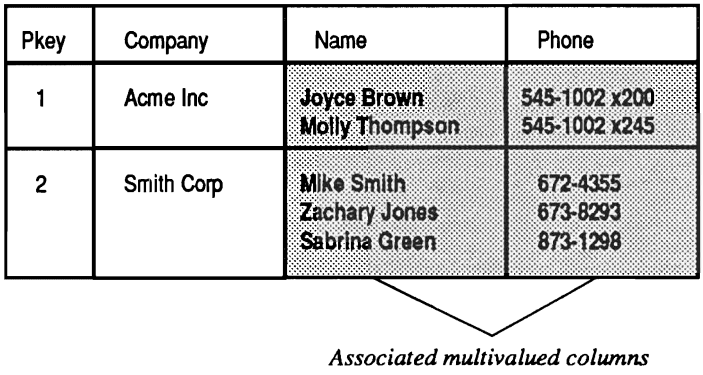
Multivalued and associated multivalued columns

You can store multiple values or repeating data in a single column by making it multivalued. For instance, you might have several contact names for an organization:

Pkey	Company	Name	City	State
1	Acme Inc	Joyce Brown Molly Thompson	Los Angeles	CA
2	Smith Corp	Mike Smith Zachary Jones Sabrina Green	Phoenix	AZ

Multiple values in one column

You can also associate two or more multivalued columns together if the repeating values have a logical relationship to each other:



About tables

You can define as many tables as you need to in Advanced Revelation. There is no limit to the number of tables you use, nor to the number of rows in each.

Valid table names

Table naming conventions in Advanced Revelation are simple. Valid table names:

- Cannot be longer than 50 characters.
- Must contain only alphabetic (A-Z) and numeric characters (0-9)
- Must be in all uppercase letters
- Must begin with an alphabetic character
- Can contain no punctuation except an underscore character (_)
- Cannot be system reserved words (SQL or R/LIST words). Examples of forbidden names include SELECT, AVERAGE, and ORDER.

Advanced Revelation will try its best to prevent you from assigning an invalid name to a table. Under some circumstances the system will suggest an alternative name based on the one you've used. In that case you can accept the suggested alternative or override with a different (but valid) name.

Tables and applications

When you create a table, it belongs to the current application. This normally means that you can only access this table while you are in that application.

A table can also be global, meaning that all applications can access the table. Alternatively, you can use a table alias (see below) to access a table from another application, as long as you have rights to access that application.

The application name is a hidden part of the table name. You can therefore create tables with the same name in different applications, and they won't interfere with one another.

Location of tables

You can keep your tables on any location that is available to your computer. This includes directories on local drives, network drives, or even CD-ROM drives.

Attaching tables

A feature of Advanced Revelation is that the location of tables is usually invisible to you. The only times you will need to know the table location is when you first create the table, and possibly when you open an application.

When you first open an application, Advanced Revelation makes the tables available to you that are on certain default locations. If you have tables on locations other than these, you must "attach" the tables on those locations before Advanced Revelation will recognize the tables.

To attach tables, you tell Advanced Revelation what location you want to attach. The system then reads an Advanced Revelation directory on that location and looks for tables that belong to the current application (or for global tables).

From that point, you don't indicate a location when accessing a table. Whenever you access a table, Advanced Revelation knows where you attached it from, and therefore knows where the table is located.

It's possible that you might have two tables with the same name on two different locations. In that case, whichever table you attached most recently is the one that Advanced Revelation recognizes under that name.

Synonyms (aliases)

In Advanced Revelation, you can establish an alias for an existing table and use the synonym table as you would the original. This is useful when you want to:

- Use a more convenient name for a table that has an awkward name.
- Have concurrent access to two tables that have the same table name.
- Have access to tables that belong to another application.

Once you have created an alias, you can use the alias in the same way you use any table. The one exception is that a synonym table name will not appear when you list the contents of a particular location, even if it is an alias for a table on that location.

Data dictionaries

Each Advanced Revelation table usually has a data dictionary associated with it. The data dictionary is a separate table that stores all the column definitions for a table. Each column you define is one entry (row) in the data dictionary.

Unlike other systems, Advanced Revelation does not use the data dictionary to structure the table. Instead, the data dictionary works as a "view" of the data in the table, in which you can add, remove, and change column definitions at any time in order to change your view of the data. Changes to the column definitions don't affect the structure of the table itself, since the dictionary is just a view of the data.

How dictionaries are used

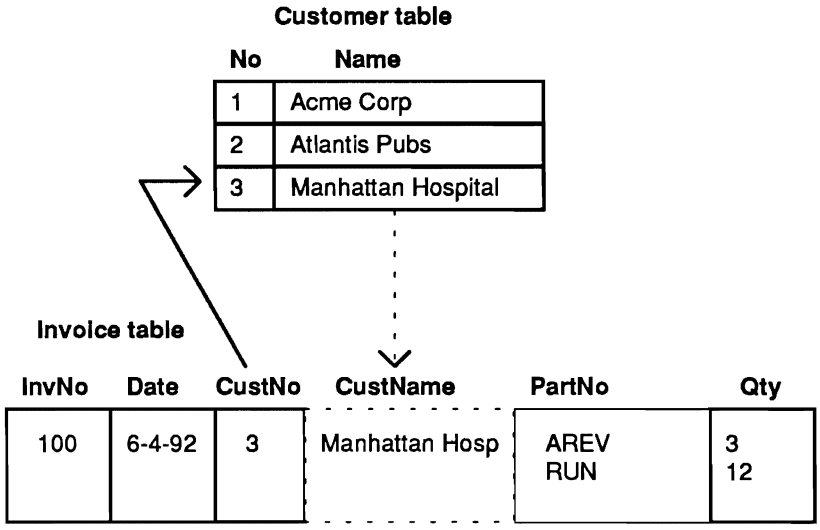
Not every table has to have a dictionary. Dictionaries are used primarily for data tables when you want to be able to access data by column names. For example, if you have defined a dictionary, you can use SQL, R/LIST, or the form or label processors to create reports, and windows to update the table. If the data in a table is not very structured—for example, in a table containing text such as programs—you don't absolutely need a dictionary. In those cases, you might still have a dictionary, but have no column definitions in it.

You can also use dictionaries when creating formulas in symbolic columns. To access data from a column, you can just use the column's name in the formula. The same is true for R/BASIC programs—you can use column names from the dictionary, including the names of symbolic columns, to retrieve data.

The dictionary for a table is itself a table. You can therefore treat it as you would any other table, and use Advanced Revelation's tools to manage the dictionary, such as creating reports, copying or deleting the dictionary, and so forth.

Related tables

Advanced Revelation allows you to create efficient and flexible databases by establishing relations between tables (sometimes called "joining" the tables). An example of a typical relation between tables is that between an invoice table and related customer table:



The customer number in the invoice table is the key to a row in the customer table. Based on this key relationship, the invoice table can "look up" the customer name as needed. The customer name is not actually stored in the invoice table.

Relations are established by storing in one table keys to rows in another table. In the example above, you store in the invoice row the key to a row in the customer table. You can then derive the related data from the customer data when querying the invoice table because Advanced Revelation can use the customer key to fetch the appropriate (related) data as necessary. *A primary key is essential to relations between tables in Advanced Revelation.*

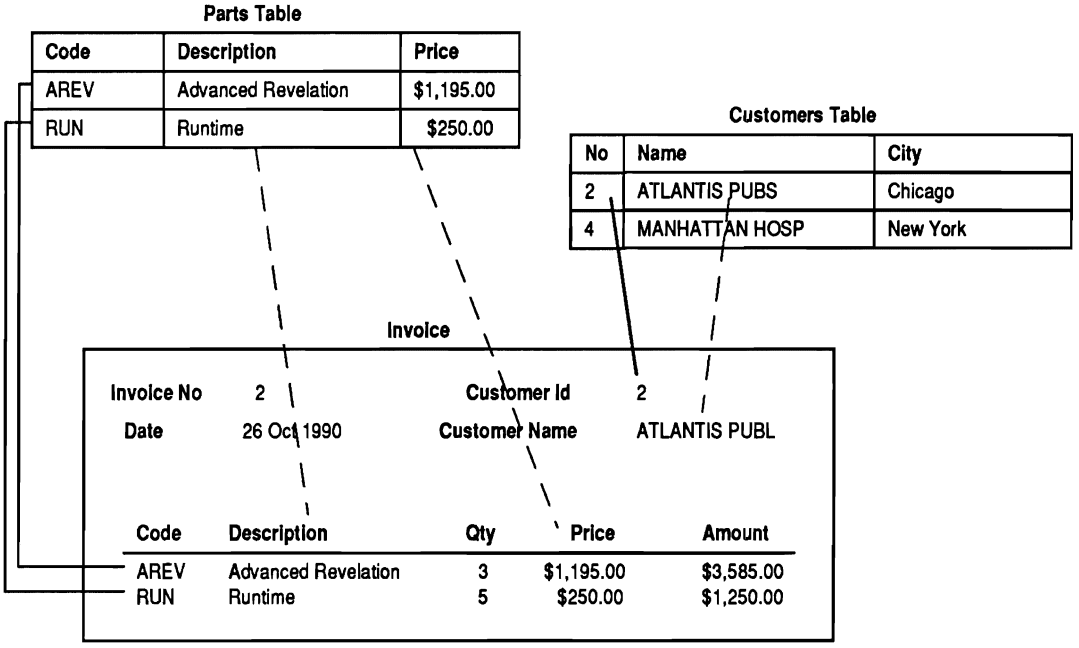
The advantages of using relations such as these are:

Space. Rather than storing all the data, you only store a key, and derive the rest of the data as needed.

Ease of maintenance. You can make an update in one place, and have it reflected throughout your system. For example, if a customer changes address, you can make the change in the customer file, and all relational references to the customer's address are automatically current.

Accuracy. Because you store the data in only one place, there is no danger of having multiple and different addresses that are difficult to keep in synch.

There is no limit to how many tables can be related together in this way. In fact, it's common to use many related tables in a database in order to reduce redundancy and to make maintenance easy. Here is another example, showing relationships between three tables:



The invoice table has relations to the products and customers table because it contains keys to rows in those tables. The dotted lines show how information from the related tables can be used in the invoices table.

When *not* to use relations

There are times when it is not a good idea to use a relation—when it is important that the data be fixed to a particular value. A good example is the list of prices in the invoice above. When you generate an invoice, you will probably derive the prices for parts from the parts table. You then create an invoice and store it away.

However, if you change the price of an item, you *don't* want all the old invoices in your tables to be updated to reflect this new price—you want the old invoices to reflect prices that were current when the invoice was created. So although you use a relation to derive the price at the time you create the invoice, you should stamp the actual price on the invoice as a permanent column. The best strategy is to use the relation to calculate a default value that you can override and that you store permanently.

Table attributes

For most uses, the default Advanced Revelation table is ideal. To accommodate unusual situations, however, Advanced Revelation lets you specify table attributes when you create the table. These help you fine-tune the performance of your tables.

You might want to change the default table attributes if:

- You know in advance how many rows there will be in your table, and you are bulk-loading the table (such as for an ASCII import). This is important only if the number of records is significant (greater than 5,000).
- Most of the rows in the table will be larger than about 2K.

For more information about table attributes, see Appendix 3, "System Administrator's guide to Advanced Revelation tables".

Tables names and users (qualified table names)

Under some circumstances a table will be tagged as belonging to a particular user. This type of table is called a "qualified" table, because its name is qualified with the owner's name. In that case, special rules apply about how that user and other users can access the table.

Qualified table names come from two sources:

- The table was created using the SQL CREATE TABLE command. The name is qualified with the name of the user who created the table.
- The table was attached from a database server using an Environmental Bond. In that case, the table is qualified in Advanced Revelation with the same user name as it is on the server—that is, with the name of the user who created it on the server in the first place (not necessarily an Advanced Revelation user name).

A qualified table name has the format:

```
username.tablename
```

where *username* is the name of the owner, and *tablename* is the tablename. Notice that the user name is separated from the table name with a period (.). That's why periods are not allowed in table names—because Advanced Revelation considers the prefix in that case to be the owner's name.

Rules for accessing qualified tables

The following rules describe how users can access qualified tables. In this discussion, "qualified" means the full table name, including the owner name and the period. "Unqualified" means using just the table name. For example, if there is a table called ALAN.CUST, the qualified version is "ALAN.CUST" and the unqualified version is just "CUST".

Using the SYS. prefix

If there is both a qualified and an unqualified table name available (e.g. ALAN.CUST and CUST), the qualified name takes precedence *for the owner*. For example, given the two tables CUST and ALAN.CUST, the user ALAN will access the ALAN.CUST

table if he uses the name CUST. So that the owner can access the unqualified table name, the prefix "SYS." has been reserved. Use this prefix in order to guarantee access to the unqualified version of a table name.

The rules for accessing tables are these:

- If you are the owner, you can access a qualified table using its unqualified name.
- If you want access to a table belonging to someone else, you access the table using its fully qualified name.
- If there is both a qualified and an unqualified version of the table available (for example, both CUST and ALAN.CUST), the qualification always takes precedence for the owner. For example, the user ALAN can use the unqualified name (CUST) to access the qualified name (ALAN.CUST), and the reserved owner name SYS to access the unqualified table (CUST, accessed as SYS.CUST).

Here is a table summarizing the rules. The top row represents a set of table names that are available *concurrently*, and the left column represents user names. The cells show you what those users would need to use as the table name to access the designated table:

user ↓ *vs table* →	CUST	ALAN.CUST	BETH.CUST
ALAN	SYS.CUST	CUST or ALAN.CUST	BETH.CUST
BETH	SYS.CUST	ALAN.CUST	CUST or BETH.CUST
MIKE	CUST or SYS.CUST	ALAN.CUST	BETH.CUST

Accessing operating system files

In addition to accessing data in Advanced Revelation tables and in tables accessible through Environmental Bonds, you can use Advanced Revelation to access data in operating system tables. For example, you can read and edit your DOS AUTOEXEC.BAT file.

Advanced Revelation reserves the table name DOS to refer to all operating system files. You can then use the full path and file name in the same way that you would use the key to a row in an Advanced Revelation table.

For example, to edit your AUTOEXEC.BAT file, you would invoke the editor using the table name of DOS, and the "row identifier" of C:\AUTOEXEC.BAT.

Note Operating system files you can access this way can be no larger than 64K.

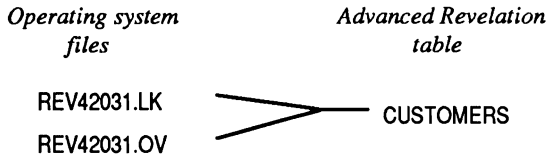
Advanced Revelation tables vs. operating system files (REVxxxxx files)

If you do an operating system DIR listing of Advanced Revelation tables, you won't see the names you are probably expecting to see. Instead you see a list of files that have the prefix REV and extensions of LK or OV, something like this:

Name	Size
REVMEDIA.LK	4096
REV42000.LK	2048
REV42031.OV	0
REV42000.OV	2048
REV42001.LK	1024
(etc.)	

In fact, these *are* your Advanced Revelation tables. However, to give you flexibility in naming tables, and to overcome limits in operating system file names, Advanced Revelation assigns names like those above to your tables. Within Advanced Revelation, of course, you see the names that you assigned yourself.

Each Advanced Revelation consists of two operating system files, one file with an extension of LK, and another with the extension OV. These two files together make up one Advanced Revelation table:



Each Advanced Revelation table consists of two operating system files.

When you create a new Advanced Revelation table, the system creates both operating system tables. The LK portion always begins with 1K in it (which is some control information), but the OV portion always begins with a size of zero.

The numeric portion of the file name is chosen randomly when the first table on a new location is created. New tables on that location use sequentially higher numbers.

Advanced Revelation stores additional information (such as column definitions and indexes) in separate Advanced Revelation tables. These are therefore additional pairs of operating system files.

The Advanced Revelation directory (REVMEDIA)

On any location where there are Advanced Revelation tables there will also be an Advanced Revelation directory. This is the operating system file REVMEDIA (with the two extensions LK and OV).

The Advanced Revelation directory stores the relationship between Advanced Revelation table names and operating system file names. Without the directory, Advanced Revelation cannot recognize the tables on a drive or subdirectory.

Because the directory is the only place where Advanced Revelation will look for a table name, you can't use operating system commands such as COPY to move Advanced Revelation tables from one location to another. Instead, you must use Advanced Revelation management tools (such as the COPYTABLE command) to do these tasks so that the Advanced Revelation directory is properly updated.

Caution The Advanced Revelation directory is always called REVMEDIA (plus LK and OV). If you copy one Advanced Revelation subdirectory over another , you will overwrite the old version of this file and lose access to all the files it described!

Environmental bonding

An important feature of Advanced Revelation is that you can get access to data that belongs to other systems, such as database servers, other database management systems (example: dBASE), or even spreadsheets or word processors. This capability is called Environmental Bonding.

When you bond to data from another source, you treat the data from that source as if it were in an Advanced Revelation table. You can generally treat the data as you would native Advanced Revelation data, and use the same tools to manage it, including data entry windows, table management tools, and the rest of the Advanced Revelation tool kit.

Occasionally there are some differences in how you must treat bonded data. As one example, Advanced Revelation allows you to change or add column definitions in a table at any time, but many other environments do not. Because of such differences, you will sometimes be referred to the section on Environmental Bonding for specific instructions on managing bonded data. If you don't find any references, you can assume that the same rules and techniques apply to bonded data as to native Advanced Revelation data.

About rows

A single entry in a table is called a *row*. As illustrated earlier in the chapter, a row is a convenient way of thinking about how entries are organized in a table. There is no limit to the number of rows that you can store in a table.

A unique feature of Advanced Revelation is that everything you create is stored as a row in a table. Obviously, all your data tables contain rows of data. But everything else you create is stored that way as well. When you create a window, it is stored as an entry in a table called WINDOWS. Column definitions are stored as rows in a data dictionary table. If you create R/BASIC programs, they are stored as rows in a table as well. Menu definitions, popups, messages, mailing label definitions, batch update definitions—everything you do in Advanced Revelation becomes a row in a table.

Because everything is a row in a table somewhere, it is important that you understand how to manage rows effectively. For example, when you learn how to copy rows from table to table, you are not just learning how to move your data, you are learning techniques that will help you move programs, windows, popups, and everything else around the system.

Row keys

Each row in a table has a column that is identified as the *key* to the row. Every row must have a key, and no two rows can have the same key. For example, in an employee table, the column identified as the row key will very likely be the employee number column. When you create a window, the name you assign to the window is the key to that row in a WINDOWS table.

Using keys for access

The key is critical to data access. Advanced Revelation uses the key as the row's address in the table. When you want to fetch or save a row, you must always provide the row key to Advanced Revelation. Because the row key is the primary means for Advanced Revelation to find a row, it is sometimes referred to as the *primary key*.

Often you will not know the row key when you want to find a row. For example, you may know an employee's name but not her employee number. In that case you can:

- Search through the table looking at all names until you have found the name you want.
- Create an index of names, and search the index for the name you want.

In both cases, the purpose of the search is to provide the primary key so that you can then get direct access to the row.

Related topics

- Key lists: select lists
- Creating indexes

Key naming conventions

There are not many restrictions on what data you can use as the key to a row. However, there are some points you should keep in mind:

- Keys may be any length up to 50 characters.
- Keys can contain almost any characters. However, we recommend that you limit row keys to the characters A-Z and 0-9 plus punctuation. No spaces are allowed. Keys are case-sensitive, so we recommend that to avoid confusion you always make your keys uppercase.
- You can use most punctuation characters in keys, but we recommend that you limit yourself to underscore (_).
- We recommend strongly against leading zeroes as part of a key. Keys are used literally, so that "1", "01" and "001" are all different keys. However, comparison operations in Advanced Revelation (for example, IF "001" EQ "01") would treat these as the same value, which affects searches and indexes.

  If your application requires keys with leading characters, we recommend that you use formatting options to display the keys with those characters as necessary.
- Do not use decimal numbers as keys. Various processes in Advanced Revelation round or truncate the decimal portions of keys, so that they are unable to distinguish keys that differ only in fractions.

Key length

Whatever the length of the key (up to 50 characters, as noted above), you should be sure that the display length in the column definition in the dictionary matches the size of the keys you will be using. Several processes in Advanced Revelation (such as the sorting routines) use the dictionary display length to determine how many characters to extract from the key. If the display length of the key in the dictionary is too short for the actual data, you could experience inconsistencies or errors when sorting, comparing, or indexing data.

If there is no dictionary definition for the column representing the key, Advanced Revelation will base its estimate on a small sample of keys extracted from the table at random. Because this small sample may not be entirely accurate, and because this sampling adds a small amount of overhead to the processes that use it, we recommend that you always establish a definition for the key column, and that you define the display length carefully.

Multipart keys

A row key can be made up of one or more columns in the row. For example, in a timesheet application you may want a row key consisting of both the employee number and the date. Each component of the key can be repeated in the table, but any particular combination of employee number and date must be unique. You can use any number of columns to define the primary key.

A note about deleting rows

When you delete a row from an Advanced Revelation table, the system overwrites the space that was formerly occupied by that row. As a consequence, you *cannot* recover a row that has been deleted. Be extremely careful when deleting a row, and be sure to have backups of critical data before attempting any deletes.

About columns

Rows in a table are represented as being divided into columns. Each column contains a discrete bit of information, such as a name, a date, etc.

Types of columns

Advanced Revelation allows you to define three different types of columns.

Data (field) columns (F type)

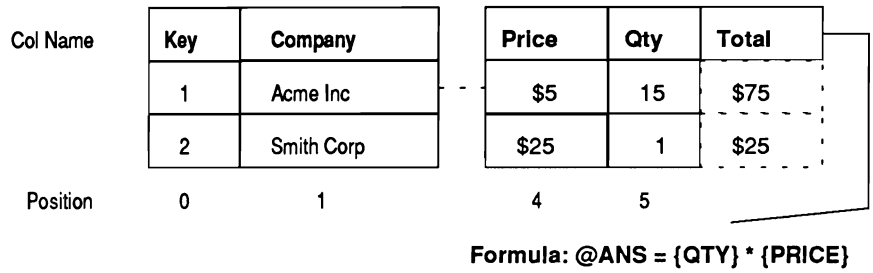
A field column is associated with a particular position in a row. The column representing the key has position zero, and the other columns in the row are simply assigned positions from one onward:

Col Name	Key	Company	City	State
	1	Acme Inc	Los Angeles	CA
	2	Smith Corp	Phoenix	AZ
Position	0	1	2	3

F-type columns are associated with positions in the row. The key column is assigned a position of zero.

Symbolic columns (S type)

Symbolic columns, also called calculated columns, are not associated directly with data in the row. Instead, they calculate a value based on a "formula". This formula can be used to calculate totals (an INVOICE_TOTAL column, for example), look up values (to extract data from a related table), or perform other processes.



S-type columns are defined using a formula. Data is not actually stored in the row.

The formula for a symbolic column can be up to 34K, and a table can contain any number of symbolic columns.

Note If you are using Advanced Revelation for OS/2, the formula for a symbolic column can be up to 64K.

Group columns (G type)

A group column is used in R/LIST reports (only) to provide a handy means of gathering several columns under a single name. It doesn't physically link the columns; instead, it gives you a shortcut way to refer to a number of columns. This is useful, for example, when you run different reports that display the same columns and don't want to keep re-entering the individual column names each time.

For example, in a STAFF_PAYROLL table you can define a single column, PAYROLL_SUMMARY, that includes every column that you need for a payroll report (EMPLOYEE_NAME, SALARY, FEDERAL_TAX, STATE_TAX, FICA, etc.). When you run the report, you can simply specify the group column PAYROLL_SUMMARY and produce a complete report without naming the columns individually.

Group columns can actually contain more than just column names. You can embed any portion of an R/LIST command into a group column and when you use that column in the report, R/LIST will substitute the contents of the group column for the group column name.

@CRT and @LPTR

Advanced Revelation creates a special group column for you when create the dictionary portion of the file. The group column @CRT contains the default list of columns to display if you specify no columns when you send an R/LIST report to the screen. You can also create the special group column @LPTR to contain the list of columns to display if you specify no columns to display when you send an R/LIST report to the printer.

Column names

Column names are subject to the same rules as table names. See "Valid table names" under "About tables" earlier in this chapter.

Master and synonym columns

When you first create an F-type column, that definition is considered the "master" column definition for that data. However, you can create a column that is a synonym of another column. There are then two definitions for the same data.

Synonym columns have two advantages: you can substitute a short column name for a long one, and you can report data in different formats (truncated from 40 to 20 characters, for example) without altering the master definition. A master column can have any number of synonyms, and each synonym can have different characteristics such as display length and justification.

Column attributes

For each F- and S-type column you can define up to 50 attributes. The most common attributes are described below.

Column name

Each column that you define in a table has a unique name (see "Column names", above). This name need only be unique within the table—you can use the same name in other tables.

Display length

Display length refers to the number of characters that appear on a report when you use a column. Display length is not a limit on the number of characters that can be stored in a column. The only limit on the length of data in a column is the maximum row size, 64K.

Note Several processes use the display length for *keys* (only) to determine how many characters from the key column to use—for example, when sorting or comparing. Therefore the display length of a key column should match the size of the actual data.

Justification

Justification indicates whether data in the column should be displayed is to be left (L) or right justified (R), centered (C), or in text (T) format (used to display paragraphs of data). The justification is also used by Advanced Revelation when it sorts or compares data.

Most typically, columns containing alphabetic or alphanumeric data are assigned a justification of L (or T), and columns containing numbers or dates are assigned a justification of R:

Mary Smith A105Y.2	**L (left)**
$1,432.00 06/07/92	**R (right)**
Welcome	**C (center)**
The quick brown fox jumped over the lazy dog.	**TN (text non-justified)**
01 Jan 92 Attached please find the requested sample. Sincerely, Sales Department	**T (text, justified)**

A text-justified column can have linebreak characters in it. If so, use T and indicate a display length of the column that matches where the linebreaks are. Text non-justified columns simply flow the entire contents of a column into a display column, breaking only on spaces.

Left, right, and centered justification truncate the display of the data (not the data itself) at the number of characters indicated by the display length. Text-justified data displays within a column the width of the display length, but for as long as necessary to completely display the data.

Effect of justification on sorting

When Advanced Revelation sorts the data in the columns, it uses the justification to determine where the significant data is in the column. All sorting is done left-justified except when the justification is R

Output format

You may choose to store certain types of data in a compact format. This is most common with dates and times, currency, and decimal numbers, but can also be used for phone numbers or other formatted data. The output format is used to specify how this data will look when it is displayed.

For example, you can enter a date as numbers (100968) and convert it to any of a variety of date formats for output: October 9, 1968, 9 Oct 1968, 10/09/68, 10-09-68, 09-10-68, etc.

Validation patterns

To ensure that data is entered properly, you can define validation patterns. As data is entered, it can be checked against a validation pattern and rejected if it does not match. For example, you could define a validation pattern that checks data entered into a phone number column. If you do not enter ten numeric characters, you will see an error message and will have to re-enter the phone number.

Data types

When you define a column, you will indicate the column's "data type". The data type—such as DATE, TIME, or DOLLARS—tells Advanced Revelation about how you want to store and display the information in a column.

Advanced Revelation has predefined display length, justification, output format, and validation patterns (described above) for twelve commonly-used data types. The default values are advisory rather than mandatory, so you can change them as desired.

Advanced Revelation Data Types

Data type	Usage	Examples
BOOLEAN	Boolean values	1/0, Y/N
CHAR(n)	Fixed-length to *n* characters	WA, NY, CT
DATE	Date	1 April 1991
DATETIME	Date and time	4/1/91 08:00:00
DECIMAL(n,m)	Fixed-point of *n* digits with *m* decimal places	0, 14.55, .25
DOLLARS (n,m)	Monetary; fixed-point of *n* digits with *m* decimal places	22.95
FLOAT	Floating-point	0, 2.18693
INTEGER	Integers	0, -4, 6424
TEXT	Variable-length (default: 4096 characters)	Four score and ...
TIME	Time	11:30:00
VARBINARY	Character strings of any value, including hex strings (default: 65,531 characters)	(no example)
VARCHAR	Variable-length (default: 65,531 characters)	Advanced Revelation, 1991

In addition to these basic data types, an Environmental Bond may offer you others that are used in foreign environments. For more information, see the chapter "Introduction to Environmental Bonding".

Extra information for data type

For some data types you are required to provide additional information. For example, if you choose the data type CHAR, you will be asked for the length. On the other hand, a data type such as DATE does not require any additional information, although you can alter the output format if you choose to.

Using symbolic columns to join tables

To establish a relationship between tables, you must have a column in one table that contains a key (or a list of keys) to rows in another column. (These keys are referred to as "foreign keys" because they are keys to a "foreign" (related) table.) Based on this data you can define symbolic columns that "join" data from the related table to the current table.

Joins are defined in the dictionary using a symbolic column. You define a symbolic join column for each column from the related table that you want to use in the current table. You can define as many join columns as you require in order to retrieve all the data you need from a related table.

For example, suppose that you have an invoice table. One of the data (F-type) columns in the invoice is the customer number, which contains the key to a row in the customer table. In the invoice you would like to see the customer's name, the address, and the phone number—each of which is a separate column in the customer table.

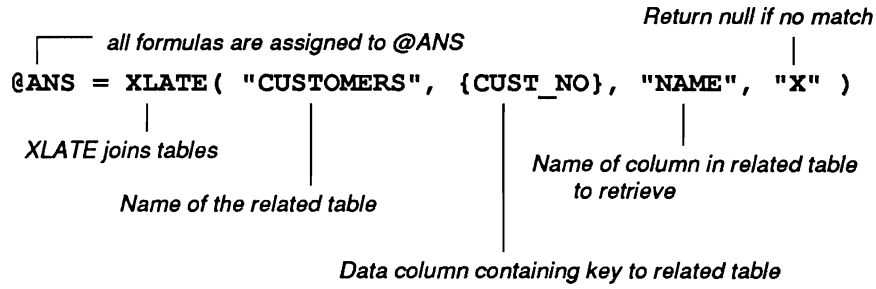
To do this, you would define three symbolic columns in the invoice table, one each for each column from the customer table that you wanted to see. In this example, you would define, in the invoices dictionary, three symbolic columns named CUSTOMER_NAME, CUSTOMER_ADDRESS, and CUSTOMER_PHONE.

Columns that join tables can be single-valued or multivalued. Examples might be:

- You have a table of ZIP codes that contains the corresponding cities. If you enter a single ZIP code into a customer row, you would define a single-valued join column to retrieve the corresponding (single) city.
- You have an invoice with a single customer number, and you want to retrieve the customer's phone number. If the phone number column in the customer table is multivalued, then the join column in the invoice table should be multivalued, too.
- You have a column in an invoice row that contains a multivalued list of part numbers, which are keys to rows in a part table. If you use this multivalued column to create a join column to retrieve part names, then the join column will return multiple part names, so it will also be multivalued.

The XLATE function

The formula used to create a symbolic column that joins to a related table uses a function called XLATE (for "translate"). Here is an instance, based on the example just above, of what the formula might look like for the symbolic CUSTOMER_NAME column in the invoices table:



Fortunately, it is easy to create a formula with XLATE by using the formula builder, which uses a window to prompt you for all the information you need. For more information, see "Building formulas" in the *R/BASIC* book.

22. Managing Advanced Revelation tables

Creating an Advanced Revelation table

The Maketable window allows you to create a new table and to define columns by filling in information at prompts. You can also use the Maketable window to change column definitions for a table that already exists.

Note The Maketable window provides a convenient place to create column definitions for your table, and allows you to see all the column definitions in a tabular form. For more precise control over column definitions, however, we recommend you use the Dictionary window to edit column definitions. See "Managing Columns" for information about the Dictionary window.

Before you use the Maketable window, you will need to know:

- Where you want the table to reside (the location of the table). As a rule, this will be a subdirectory or drive name. The subdirectory or drive must already exist and be available to your system.

  Note If you don't put the table on a default location, you will need to attach the drive or subdirectory when you next open the current application, or Advanced Revelation won't be able to find the table.

- Whether you want to indicate any special attributes for the table. Unless you need to fine-tune the table for performance, you should accept the default attributes for the table. For more information on table attributes, see "About database in Advanced Revelation" and the appendix "System administrator's guide to Advanced Revelation tables".
- The names and characteristics of any columns you want to define for the table. However, you can add and modify these column definitions later if you prefer.
- Which column is the key column (or which ones, if you want to define a multipart key). You will need to define the key column(s) first.

To create a new table, follow these steps:

1. Choose the menu option **DB Admin-Table-Create**.
2. At the *Table name* prompt, enter the name of the table. If you enter an invalid name (see "Valid table names", above), the system will suggest an alternative.
3. At the *Location or volume p*rompt, enter the drive or subdirectory name where you want the table to be located.
4. At the *Table attributes* prompt, press [F2] or click on <Options> if you want to indicate any special attributes for the table. We recommend that you accept the default attributes and simply skip over this prompt.

5. When you are finished at the *Table attributes* prompt, Advanced Revelation will create the dictionary portion of the table. If you don't want to define any column definitions, press [F9] or click on <Save>, and Advanced Revelation will create an empty data table and an empty dictionary.

Creating basic column definitions in the Maketable window

To create basic column definitions for the new table, follow these steps:

1. At the *Column name* prompt enter the name of the column.
2. If this is the first column you are creating for this table, the system will ask you if it is the key column. Indicate "Yes" to create the key column.
3. The system will fill in the *Type* prompt with "F" (field) and the *Pos* column with the next column position. and skip to the *Datatype* prompt. Enter the correct datatype for the column you are defining. Press [F2] or click <Options> to see a list of available data types.
4. If you want to modify the values for the datatype you have chose (example: the precision of a floating-point data type), press [F2] or click <Options> after you have selected a datatype and while the cursor is still in the *Datatype* column. A window will display with the attributes for that datatype, which you can alter as necessary. We recommend that you simply accept the default values associated with your selected datatype.
5. After you have finished with the datatype, the prompts for *Col heading, Len* (length), *J* (justification), and *S/M* (single or multivalued) are filled in automatically. If you want to change the values here, move the cursor to the attribute you want to change, press [F4] to turn the editor on, and enter your new value.

Creating symbolic columns in the Maketable window

If you want to create a symbolic column definition while defining columns in the Maketable window, follow these steps:

1. At the *Column name* prompt, enter the column name for the symbolic column.
2. The cursor will skip to the *Datatype* column. Make sure first that the cursor is block-shaped (press [F4] if it is not), then use the [←] key to move back to the *T* (type) column. Enter "S" for symbolic.
3. The formula window will appear. Enter the formula for this symbolic column, and press [F9] or click on <Save> when you are done.

4. You will return to the *Datatype* column of the Maketable window. Enter a datatype appropriate to the data that your formula will calculate.
5. From this point you can continue defining the column as you would any other column.

Defining a multipart key

To create a multipart key, follow these steps:

1. Create the first part of the key by entering in the name of the column at the *Column name* prompt.
2. When the cursor moves to the *Datatype* prompt, make sure the cursor is block-shaped (press [F4] if not), then press [←] to move back to the *Pos* prompt. Enter a key position number at the prompt:

   ```
   0*1
   ```

   Where "0" means that the column you are creating is a part of the key and "1" indicates that it is the first part of the key.
4. Finish the rest of the prompts as you would for any other column.
5. Enter in the second (and subsequent) parts of the key by repeating steps 1 through 4. However, at the *Pos* prompt, enter in the appropriate key position information. For example, the two key position values:

   ```
   0*2
   0*3
   ```

   represent the second and third parts of a multipart key, respectively.

Defining a synonym for a column

After you have defined an "F"-type column, you can define as many synonyms for that column as you like. To define a synonym column, follow these steps:

1. Place the cursor on the column for which you wish to define a synonym (the "master column").
2. Press [Shift-F8] or click on <Softkeys> then Make Synonym.. A new row will open immediately following the column for which you are defining a synonym.
3. When prompted, enter in the synonym name.
4. A " (quote mark) will appear in the *T* prompt to indicate that this is a synonym. It will also copy the position of the master column to the *Pos* prompt for the synonym column.

5. Finish the rest of the prompt as you would for any other column.

Note The data type of a synonym does not have to match that of the master column.

The position (the *Pos* prompt) of a synonym always matches that of the master column. The master column is always listed first in the Maketable window, followed by any synonyms that you have created for that column.

Defining a group column

To create a group column, follow these steps:

1. Enter a name for the group column at the *Column name* prompt.
2. After you have entered the name, press [Shift-F7] or click on <Softkeys> then Group definition to tell the Maketable window that you want to create a group column.
3. You will see a message that prompts you to confirm that you want to convert the column from an "F" to a "G"-type. Press [Enter] or click on <Ok> to confirm.
4. A Group window appears. Fill in the columns that constitute the group. You can also add selection and sort criteria for a report.
5. When you return to the Maketable window, notice that the number at the *Pos* prompt disappears, since group columns are not associated with a row position. The rest of the prompts for that column will be skipped, since they do not apply to group columns.

Editing the Group

You can view or change the entries for a group column at any time. To do so, place the cursor on the group column you wish to edit, and press [Shift-F7] or click on <Softkeys> then Group Definition.

Attaching a table to your system

When you first log on to Advanced Revelation, the system attaches a default set of tables. To get access to tables that are on locations other than the default:

1. Choose the menu option **DB Admin-Tables-Attach.**
2. At the *Location* prompt enter the path or location on which the tables you want are located.
3. If you want to attach only a subset of tables on that location, enter the name(s) of the table(s) at the *Table names* prompt. Press [F2] or click <Options> to see a list

of tables available at that location (you won't see the any alias names in this listing).

Note You can configure your system to attach tables on a default location , or to execute a logon script that attaches tables from a series of locations. See the chapter "Managing your system".

Related topics

- TCL ATTACHTABLE and SETATTACH commands
- Configuring the workstation (Startup commands)

Detaching a table

When you detach a table, the table still exists, but is not available for use in your application. If you want to attach a table with the same name as an available table, for example, you need to detach the first table before the system will allow you to attach the second one.

You can detach single tables, or all the tables on a particular location. To detach a table:

1. Choose the menu option **DB Admin-Tables-Detach.**
2. If you want to detach all the tables on one location, at the *Location or volume* prompt, enter the subdirectory or path name on which the table to be detached resides. If you want to detach specific tables, skip this prompt. Press [F2] or click <Options> to see a list of all the locations currently attached to your application.
3. If you want to detach specific tables, at the *Table name(s)* prompt, enter the names of specific tables to detach on that location. If you want to detach all the tables on one location, skip this prompt. Press [F2]or click <Options> for a list of attached tables.

Related topics

- TCL DETACHTABLE command

Clearing a table

When you clear a table, you delete all the rows in the table, but keep the (empty) table available and the table structure intact. By default Advanced Revelation will clear only the data portion of the table. If you want to clear both the data and the dictionary portions of the table, repeat the process twice, once for each portion.

Caution! You cannot recover rows after clearing a table, so be sure to back up or take other cautionary steps before clearing the table.

To clear a table:

1. Choose the menu option **DB Admin-Table-Clear.**
2. At the *Table name* prompt, enter the name of the table to clear. If you want to clear only the dictionary portion of the table, enter the word DICT followed by a space before the table name.

Deleting a table

When you delete a table, you completely remove it from your system. By default, Advanced Revelation will delete both the data and dictionary portions of the table.

Caution! You cannot recover a table that has been deleted, so be sure to back up or take other cautionary steps before deleting valuable data.

To delete a table:

1. Choose the menu option **DB Admin-Tables-Delete.**
2. At the *Table name* prompt fill in the name of the table to delete. Press [F2] or click <Options> to see a list of available tables.

To delete only the data or dictionary portion of the table:

1. Choose the menu option **DB Admin-Tables-Delete.**
2. At the *Table name* prompt, fill in the table name preceded by the word DICT (to delete only the dictionary portion) or DATA (to delete only the data portion). Separate the word DICT or DATA from the tablename using a space.

Related topics

- TCL DELETETABLE command

Copying or moving a table

Copying a table makes an entirely new copy of it in a new location. Moving the table is the same operation, but you delete the original once the copy has been completed.

Caution! You cannot recover a table that has been deleted, so be sure you've backed up or taken other steps to preserve valuable data.

When you copy a table in Advanced Revelation, you have these options:

- Copy the table to a new location or to the same location under a new name.
- Create a new table during the copy process or overwrite an existing table.
- Rename the table as you are copying it.
- Copy a table only if the destination table exists.
- Copy the table to a new application (as long as that application already exists), with or without renaming the table at the same time.
- Specify that you want to copy only the data or the dictionary portion of the table.

When you have finished the copy operation, the destination (new) table will not be available. To have access to the destination table, you must first attach it.

About copying an indexed table

If the source table has been indexed using a Btree, Cross Reference, or Quickdex/Rightdex index, the destination table will retain the indexes. However, if the source table has a relational index, you can move the table (deleting the source) but you cannot copy it. Your choice is to copy the table with the delete option, or to remove the relational index before attempting to copy the table.

Copying the table

1. Choose the menu options **DB Admin-Table-Copy**.
2. At the *Source table* prompt, enter the name of the table to copy. Press [F2] or click <Options> for a list of available tables. If you want to copy only the dictionary portion of the table, enter the word DICT (followed by a space) before the table name. If you want to copy only the data portion of the table, enter the word DATA (followed by a space) before the table name.
3. At the *Location* prompt, enter the subdirectory or path name to which you want to copy the table. Leave this blank if you are copying the table to the same location under a new name or application.
4. At the *Application* prompt, enter the name of the application to which you are copying. Skip this prompt if you want to copy the table to another table in the same application.
5. At the *Destination table* name prompt, enter the name of the destination table.
6. At the *Options* prompt, enter the code for any option you want. Press [F2] or click <Options> for a list of valid options. If you are moving the table instead of just copying it, enter "D" (delete source) here.

The table you have copied will not be available until you attach it.

Related topics

- Indexing
- TCL COPYTABLE command

Renaming a table

Renaming a table assigns a new name to the table. You can optionally assign the table to a new application at the same time. If you rename the table within the current application, it is immediately available to your system under the new name.

Note Renaming the table does not update any references (in programs, symbolic column definitions, etc.) to the old name-you are responsible for making those changes.

If there is a relational index on the table to be renamed, then the related table (the target of the index) must be available at the time you rename the table.

To rename a table:

1. Choose the menu option *DB Admin-Table-Rename*
2. At the *Current table* prompt, enter the name of the table to rename. Press [F2] or click <Options> to see a list of available tables.
3. If you want to assign the table to a new application or want to make the table GLOBAL, enter the appropriate information at the *New application name* prompt. (See below).
4. At the *New table name* prompt enter the new table name.

Related topics

- TCL RENAMETABLE command

Assigning a table to another application

You can transfer a table from the current application to another application, as long as the second application already exists. To do so, follow the instructions for renaming a table, but at the *New application name* prompt, enter the name of the application to which you want to assign the table. If the other application has a password, you must know that password. When the operation is finished, the table will belong to the second application, and will no longer be available to the current application.

Making a table globally available

You can make a table available to all applications by making it a global table. To do so, follow the instructions for renaming a table, but assign it to the application GLOBAL.

Related topics

- TCL RENAMETABLE command

Listing available tables

To see a list of all tables currently available, use the menu option **DB Admin-Table-List**. Enter "P" at the *Options* prompt to send the report to the printer. The listing shows you:

- The name of the table.
- What application it belongs to (current or GLOBAL).
- What location it resides on.
- What type of table it is.

Related topics

- TCL LISTTABLES command

Listing all tables on one location

Listing all the tables on one location is like doing a directory listing of that location. The resulting report shows you all the tables that have been created on that location, even if they are not available to the current application. To get a directory listing of all the tables that reside in one location:

1. Choose the menu option **DB Admin-Location-List**.
2. At the Location prompt enter the location for which you want a listing.
3. At the Options prompt, enter "P" to send the report to the printer.

The listing shows you:

- The name of the table
- The account to which the table belongs

- The name of the table at the operating system level; that is, the DOS or OS/2 file that stores the data for that table
- Any MFSs (Modifying Filing Systems) installed for that table. MFSs of particular interest are:

SI.MFS	The table is indexed using a Btree, Cross Reference, or Relational index.
PROTECT.MFS	The table has control features installed, meaning it is eligible for SQL and for Transaction Processing.
QUICKDEX.MFS	The table has a Quickdex index installed.
RIGHTDEX.MFS	The table has a Rightdex index installed
PROGRAM.MFS	The table contains R/BASIC source code entries (the object code will be put into a table called OBJECT).

Related topics

- TCL LISTVOLUME command
- Indexing

Creating an alias (synonym) for a table

An alias allows you to use a more convenient name for a table. It also gives you access to tables that belong to other applications. You can make the alias temporary (only good for the current session of Advanced Revelation) or permanent (reusable in subsequent sessions).

If you create a permanent alias, you can access it in subsequent sessions by attaching the location (see "Attaching a table" above) and specifying the alias name as the name of the table to attach.

To create an alias:

1. Choose the menu option **DB Admin-Table-Setalias.**
2. At the *Location* prompt, enter the subdirectory or drive name where the table is located for which you are creating an alias.
3. At the *Application* prompt, enter the name of the application to which the table belongs. Skip this prompt if you are creating an alias for a table in the current application.
4. At the *Table name* prompt, enter the name of the table for which you are creating an alias.

5. At the *Alias name* prompt, enter the new alias name.
6. At the *Options* name, enter "W" (write) if you want to save this alias name as a permanent alias.

Related topics

- TCL SETALIAS command

Accessing tables in another application

If you have permission to access another application, you can also access the tables from the application from within your current application. If the application to which the table belongs has a password, you must know the password.

To access tables from another application, follow the steps for creating an alas for a table. However, at the *Application* prompt enter the name of the application to which the table belongs. When you've saved the synonym name for the table, it will be available to your current application.

23. Managing rows

Managing rows

Deleting rows

To delete rows in a table, follow these steps:

1. Choose the menu option **DB Admin-Row-Delete**.
2. At the *Source table* prompt enter the name of the table from which to delete rows. Press [F2] or click on <Options> for a list of available tables.
3. At the *Row(s)* prompt enter the row keys of the rows to delete.
4. At the *Options* prompt enter "A" to display row keys as the row is deleted, or "S" to suppress all messages.

Caution You cannot recover a row that has been deleted, so proceed cautiously.

Related topic

TCL DELETEROW command

Copying rows

Advanced Revelation enables you to copy rows within a table—from one key to another—and between tables. Because rows in an Advanced Revelation table are not rigidly structured, you don't need to worry that a row from one table won't "fit" into another table.

In addition to simply copying rows from one place to another, you can choose several copy options:

- **Overwrite.** Without this option, Advanced Revelation will not allow you to overwrite a row that already exists. Overwriting is also faster, because Advanced Revelation doesn't have to check first whether an old version of the row exists in the destination table.
- **Delete.** Removes the source row after it has been successfully copied to the new location.
- **New row inhibit.** If you are explicitly overwriting rows, the new option helps prevent typing mistakes by making sure that you only copy rows that already exist in the destination location.
- **Lock.** Locks the source and target rows (on a network) before copying.

To copy rows within or between tables, follow these steps:

1. Choose the menu option **DB Admin-Rows-Copy**.
2. At the *Source table* prompt, enter the name of the table where the row is currently stored. Press [F2] or click on <Options> for a listing of available tables.
3. At the *Destination table* prompt, enter the name of the table to which you want to copy the row. If you are simply copying it to a new name in the same table, skip this prompt (it will be filled in automatically).
4. At the *Source row(s)* prompt, enter the keys of the rows you want to copy. Press [F2] or click on <Options> to see a popup of available rows.
5. If you wish to assign a new row key to each row you are copying, enter the new name at the *Destination row(s)* prompt. The rows will be copied from the source names to the destination names with a one-to-one correspondence. If you have fewer destination names than source names, the extra source rows will be copied using their original names. If you don't intend to rename the rows, use [↑] to move up out of this prompt.
6. At the *Options* prompt, enter a letter to indicate any copy options. Press [F2] or click on <Options> for a list of available options.
7. Start the copy process with [F9] or by clicking <Save>.

Related topics

TCL COPYROW command

Renaming rows

Strictly speaking, you cannot rename a row. Instead, you make a copy of it under a new key. Follow the steps for copying a row, but be sure to:

1. Enter a new name for the row at the *Destination row(s)* prompt.
2. Use the "D" (delete) option so that the original row is removed after the new one is created.

Printing or displaying rows

If you want to print a row on the printer or display it on the screen, you can use the Copyrow process. Follow the steps for copying a row, but be sure to:

1. Skip the *Destination table* prompt.
2. Skip the *Destination rows* prompt. Use [↑] to move out of this prompt.
3. At the *Options* prompt, choose "P" to print to the printer, or "T" (terminal) to display the row on the screen. If you select multiple rows to copy, there will be a

page break between each one. Choose the option "K" to suppress (kill) the line numbers.

Using row management tools with operating system files

You can use Advanced Revelation's row management tools to maintain and manage operating system files as well. All the usual row management tools are available: edit, copy, delete, etc. However, the operating system files that you manage from within Advanced Revelation have the same restrictions as any other row—that is, they cannot exceed 64K in length.

To access operating system files, use the table name DOS. Use the operating system path and file name at any place that you would normally enter the key to an Advanced Revelation row. For example, you can edit your AUTOEXEC.BAT file using the Advanced Revelation editor. Just indicate the table name of DOS and the "row" of C:\AUTOEXEC.BAT.

You may also need to convert between Advanced Revelation and operating system formats. The Advanced Revelation Copyrow process and the editor have these options to convert between formats:

- **Revelation format.** Each line in the row is terminated with Advanced Revelation's end-of-column character (ASCII 254).
- **ASCII/DOS format.** Each line in the file ends with a carriage return-line feed combination, and the file is terminated with Ctrl-Z (ASCII 26).

If you are copying an Advanced Revelation row to operating system format for use by non-Advanced Revelation systems (example: a word processor), then convert the row to ASCII format. However, if you intend to use the row in another Advanced Revelation system, copy the row in Revelation format.

Copying Advanced Revelation rows to operating system files

Copying an Advanced Revelation row to an operating system file is useful when you want to transfer it to another Advanced Revelation system. This is common, for example, in transferring programs between systems.

Note If you want to copy more than one Advanced Revelation row into a single operating system file, use the Export process. Copying using COPYROW creates a single DOS file for each Advanced Revelation data row.

To copy an Advanced Revelation row to an operating system file, follow the steps for copying a row, but be sure to:

1. Use the table name DOS as the destination table name.
2. Provide an operating system name, including the full path, at the *Destination row(s)* prompt.
3. Choose the option "I" to convert the row to ASCII format if the resulting operating system file will be used by a non-Advanced Revelation application.

Copying operating system files to Advanced Revelation rows

To copy a complete operating system file to into an Advanced Revelation table as a single row, follow the steps for copying a row, but be sure to:

1. Use the table name DOS as the source table name.
2. Provide an operating system file name, including the full path, at the *Source row(s)* prompt.
3. Choose the option "R" to convert the row to Revelation format if the operating system file is formatted with a carriage return-linefeed after every line.

Updating rows using batch update

Advanced Revelation provides a facility for updating a number (a batch) of rows in a table in one process.

About batch update

You can change rows in a table by using an application window. If you have to change a large number of rows, however, this can be tedious and prone to errors. By using batch update, however, you can make uniform changes to all (or a selected group) of rows in a table.

Batch update uses formulas to update rows. Therefore, in order to use batch update effectively you will need to be familiar with how to build formulas. Some examples are given below, but for more information, refer to the chapter on formulas in the *R/BASIC* book.

Using batch update

Batch update works in four steps:

1. Create a batch update definition that describes the change you want to make. The change is described using a formula. You can define as many different batch update definitions as you like, and can reuse them as many times as you like.
2. Run the batch update process to create the changed data rows in a transaction (temporary) table. Because changes aren't written immediately to your data table, you enjoy a high degree of database protection.
3. Examine the changes to make sure they are what you wanted.
4. Integrate the changes back into the original data table.

About batch update formulas

For each column that you want to update, you must provide a formula that describes the change. Formulas can be small or large, simple or complex. The important thing, however, is that at the end the formula produces a changed value for the column.

The basic outline of a formula is that you calculate or modify a value of some sort, then assign that value to the keyword "@ANS". The value you are calculating can use any information that's available to your application, no matter where it might be. For example, suppose you wanted to change the value of a column called DISCOUNT by adding 15% to the value already there. The formula would look something like this:

```
@ANS = {DISCOUNT} * 1.15
```

This extracts the current value of the column DISCOUNT, adds 15% to it, and puts the resulting value into the keyword @ANS.

If you assign this formula to the column DISCOUNT, the batch update process will overwrite the current value that is in that column with the value you've calculated. You could also use this calculated value to update a different column in the row—it's entirely up to you.

You can make multiple changes to the row, even if the changes are dependent on one another. For example, suppose that you have a table in which the name information has been entered in the format *lastname, firstname initial*. Perhaps you've decided that it's more practical for you to have the last and first names in different columns. You could write two formulas, one each to update a (new) LASTNAME column, and the second to update the original NAME column. These two formulas extract the relevant portion of the name:

Column to update	Formula
LASTNAME	`@ANS=FIELD( {NAME}, ",", 1)`
NAME	`@ANS=FIELD( {NAME}, ",", 2, 3 )`

Formulas can be more than one line. In fact, they can be up to 34K long. You can use any R/BASIC command in a formula, including IF-THEN-ELSE, subroutines, or anything else you require.

The formula builder

When you define the batch update process, you use the Update Formula window. If you want help in building the formula, you can invoke the Formula builder by pressing [F2] or clicking on <Options>. This displays the formula assistant window. To get help, press [F2] or click on <Options> again.

For more information about using the formula window, see the *R/BASIC* book.

Important notes for building formulas

However you choose to build your formulas, keep these important points in mind:

- *You must assign a value to @ANS or the result will be a null value.* This is easy to overlook when you use IF. Look at this formula that assigns a new salesman name to all the rows in which the state is NY:

```
IF {STATE} EQ "NY" THEN
   @ANS = "Steve"
END ELSE
   @ANS = {SALESMAN}
END
```

  Note that no matter what the condition is, a value is always assigned to @ANS.

- *Formulas are processed from the top to the bottom*—the first formula in your definition first, the second one second, etc. It's important to remember this if you are creating formulas that depend on other columns (as in the example about LASTNAME and NAME, above). If the first formula updates a value that is used in a subsequent formula, the most recent one (including the update) is the value that will appear in the second formula.

Building the batch update definition

Use the Batch command window to construct the update process definition. Before you build a batch update definition, you should:

- Know a name that you want to assign to this update definition.
- Have a transaction (temporary) table available to hold the update transactions until you integrate them. This table should be empty.
- Know the formulas you need to update the affected columns.

To build the definition:

1. From the Development menu, choose **Developer-Batch-Batch.**
2. At the *Update name* prompt, enter the name you want to assign to this batch update definition.
3. At the *Table name* prompt, enter the name of the table you want to update.
4. At the *Transaction table* prompt, enter the name of the temporary table. This table must already exist. If this is not a new table, you must clear the temporary table using [Shift-F7] before running the batch update.
5. At the *Column to update* prompt, enter the name of the column to update. As soon as you are done entering the column name, the Update Formula window appears.
6. At the Update Formula window, enter the formula that describes the update you want to make. Press [F2] or click on <Options> to display the Formula window, then press [F2] or click on <Options> again to invoke the formula builder.
7. Repeat steps 5 and 6 for any other columns you want to update.
8. Save your update definition with [F9] or by clicking <Save>.

Running the batch update

Once your batch update definition is created, you can run the update. Re-display the Batch window (the one used to create the batch update definition) and call up your batch update definition. If necessary, clear the transaction table using [Shift-F7]. Then press [Shift-F1] to start the update.

The batch update processor will then update each row in the table, and place an updated version of the row into the transaction table.

Updating selected rows

If you want to update only a selected list of rows, use the Filter key ([Ctrl-F10]) to activate a select list before starting the update. From this Filter Selection popup you can choose to select certain rows for update, or to re-call a list of keys stored earlier. After activating the list of keys, press [Shift-F1] to start the update.

For more information about the use of key lists, see the information on the Filter key in the chapter "Using windows for queries".

Checking updates

When the update is through, check the transaction table to be sure that the changes are correct. To do so:

1. Choose the option **Developer-Batch-Compare.** The Batch Compare window displays.
2. At the *Name given to update process* prompt, enter the name of the batch update definition for which you want to check the results. Press [F2] or click on <Options> to see a list of definitions.
3. Indicate whether you want to send the report to the printer or the screen (terminal).
4. Examine the results. The original version of a column will be listed under the a heading with the prefix "OLD_". The new version will be under the column name.

If the changes are as you had intended, you can proceed to integrating the changes. If there are problems, however, you have some choices:

- Change the batch update definition and re-run the update.
- Make individual changes in the transaction table. This would be practical if there are only a few problems in an otherwise successful update run.

If you choose the latter option, you can use any edit tool (the editor, a window, etc.) to make the change.

Integrating the changes

If the changes are as you want them, you can integrate them back into the original table. To do so, choose **Developer-Batch-Integrate,** then at the Ingrate window fill in the name of the batch update definition for which you are integrating changes. The integration is done immediately.

Caution You cannot undo the changes once you have integrated them. Be sure that the changes have been successful, and that you have a backup copy of the original table in case of errors.

24. Managing columns

About creating or modifying column definitions

You have several options for creating and modifying column definitions:

- Use the Make Table window. This is a convenient way to define columns at the same time that you are creating a new table.
- Use Paint to bond prompts in a window to columns. Paint gives you the option to create new columns or modify existing ones.
- Use the Dictionary window to create and modify column definitions individually.

Although all of these methods work well, using the Dictionary window can be more convenient, because it is faster and allows you to see (and modify) more attributes for each column. The instructions below assume that you will use the Dictionary window

About the effect of modifying column definitions

Adding or altering a column definition has *no effect on the data table.* When you alter a column definition, you change your "view" of the data table—either by adding a new way to look at a column, or by changing the way an existing column definition treats the data. However, there is no structural change to the table.

For example, if you add a column for a row position that currently is undefined, Advanced Revelation does not restructure the table to accommodate the new position. If you create a report using the new definition, the column will simply report a null value, since there is no data at that position. If you update the table using the new column definition—using SQL, say, or by adding a new prompt to a window—then the new data is added to the rows that you edit in that way. Again, however, there is no restructuring of the table, only of the rows that you update. Because of Advanced Revelation's variable-length row structure, the rows simply grow or shrink to accommodate exactly the amount of data that you store in them.

About column definitions and prompts in a window

Prompts in a window and column definitions in a dictionary have many attributes in column—display length, row position, output formats and validation patterns, etc. However, each is stored separately. Column attributes are stored in the dictionary, while prompt attributes are stored in the window definition.

Because prompt and column attributes are stored separately, changes you make in the dictionary do not affect windows that already exist. If you create new windows from the changed column definitions, then of course those changes will be reflected there. There is an option in Paint to update prompts to reflect new column definitions (and vice versa), if you do want to change a window to match the updated dictionary.

About multivalued column definitions

A column that is defined to be multivalued does not in and of itself put multiple values into a column. As with everything else in the dictionary, it simply instructs Advanced Revelation in how to treat the data during a report or in a (new) window.

The most important effect of defining a column to be multivalued is in the way that the column is used for sorting. In R/LIST reports, a multivalued column will cause an "exploding sort". In SQL, a multivalued column will cause the row to be "normalized". In both cases, this means that the report will be created as if each value in the multivalued column were a row by itself, with the other columns in the row being repeated as many times as necessary. For more information about exploding sorts, see the keyword "BY" in the chapter "R/LIST keyword reference" in the *Reference* book. For information on how SQL normalizes columns, see "Using columns with multiple values" in the chapter "An introduction to SQL".

At the same time, if you *do* have multiple values in a column that is defined to be single-valued, the data will display improperly in a report. A typical symptom is that you see superscript 2 characters (^{2}) in your report.

Prerequisites for creating or altering column definitions

Before you create or modify column definitions, you should know the following. Unless otherwise noted, this information is discussed in detail in the chapter "About databases in Advanced Revelation".

- Know about column naming conventions.
- Know whether you are creating a data column (called a field column, type F), a symbolic column, or a group column.
- Know if this is a key column, and if so, whether it is part of a multipart key.
- Know whether the column will be containing a single value or multiple values.
- Know about data types and how Advanced Revelation uses data types to determine output formatting for a column.
- Know if you want or need to apply an output format to the column. Output formats (and validation patterns) are described in detail in the chapter "Creating windows using Paint".
- Know if you want to apply any validation patterns to the column.
- If this is a data column, know what position in the row this defines. For a report of the existing column definitions (and the positions they define), see "Listing column definitions" later in this chapter.

- If this is a symbolic column, know the formula that will define the column. For more information about formulas, see also the chapter "Building formulas" in the *R/BASIC* book.
- If this is a group column, know what other columns this will be grouping together.
- Know what justification you want to apply to the column. Selecting the correct justification is important if you are defining keys, if you will be sorting the table using this column, or if you will be indexing this column.
- Finally, know what display length you want to assign to this column. Display length can be changed at any time, but is critical for sorting key columns.

Managing columns

Creating a new data (F-type) column

Please read "About creating or modifying column definitions", above. To create a new F-type columns:

1. From the Development menu, choose **DB Admin-Columns-Define**. The Dictionary window displays.
2. At the *Table name* prompt, enter the name of the table for which you are creating a column definition. Press [F2] or click <Options> to see a list of available tables.
3. At the *Column name* prompt, enter a new column name.
4. At the *Dict type* prompt, enter "F" to indicate that you are creating a data (field, F-type) column.
5. At the *Single/Multi* prompt, enter "S" if the column will be single-valued, "M" if it will be multivalued. If you are defining a key column, this *must* be a single-valued column.
6. At the *Data type* prompt, enter a data type that describes what sort of data this column will contain. This prompt is optional. For a list of available data types, press [F2] or click <Options>.
7. If the data in column will be stored in an internal or compressed format, enter at the *Output format* prompt a format specification that will be used to convert the data to its proper display format. Press [F2] or click <Options> for a list of commonly-used formats.
8. At the *Validation patterns* prompt, enter one or more validation patterns that Advanced Revelation should use to make sure that entries into this column are valid. Press [F2] or click <Options> for a list of commonly-used validation patterns.

9. At the *Position* prompt, enter the position within the row that this column will define. The default value here is the next available position in the row. If you are creating a definition for the row key, the value here should be zero.

   If you are creating a multipart key, enter at the *Part* prompt what portion of the key this column defines. For example, if the new column is the second column in a three-part key, enter "2" here.

10. At the *Column heading* prompt, enter a heading that should be used in R/LIST and SQL reports. If you want to have a heading that is on two or more lines, enter each line of the column heading on a separate line of this prompt.

    Note If the column heading is longer than the display length you assign to the column, the length of the column heading will take precedence.

11. At the *Justification* prompt, enter a code for how the column should be justified for R/LIST and SQL reports. Press [F2] or click <Options> for a list of justifications.

    If the data will be a date or number, choose "R". If the data in the column will be a long text strings, choose "TN" (text, non-justified). If the data will contain hard line breaks, choose "T".

12. At the *Display length* prompt, enter a number for how wide the display will be on R/LIST and SQL reports. If this is a text column, enter in the value for how wide the column should be before the data is wrapped to the next line.

    Note If you are defining a key column, enter a display length that is an accurate reflection of the length of the data, because internal processes in Advanced Revelation will use this length to determine how many characters in the key are significant. See "About rows" in the chapter "About databases in Advanced Revelation".

13. At the *Description* prompt, enter any amount of text that describes this column. The information here can be used as a help message in a window, and can be used in reports listing all the columns in a table.

14. Save your changes with [F9] or by clicking on <Save>.

Modifying a column definition

Please read "About creating or modifying column definitions", above. You can modify any attribute of a column at any time. However, be aware of the following:

- Because changing the definition of a column does not affect the data, you cannot modify a column and alter your table. For example, you cannot change the position value of a column and have it shift the data in the row. If there is already

data in the table, you cannot have date formatting apply to it afterwards by changing the output format of a column definition.

- If you change the output format or validation pattern, these changes will be reflected the next time you build a window or update the table using SQL. However, the changes do not affect existing data or existing windows (windows store their own version of these values when they are saved).
- If you alter the justification of a column that is indexed, the index will not reflect the change in justification unless you remove and re-install the index.

To modify a column definition, display the Dictionary window (see "Creating a new column definition", above) and enter the table and column name for the existing column. Then move to the prompt you want to modify and enter a new value. Save your changes with [F9] or by clicking <Save>.

Deleting a column definition

Please read "About creating or modifying column definitions", above. Deleting a column definition does not remove the data associated with the column. If the column you delete was indexed, Advanced Revelation removes the index information as well.

To delete a column definition, display the Dictionary window (see "Creating a new column definition", above) and enter the table and column names for the column. Then press [Alt-D] twice to delete the column definition.

Caution! Deleting a column definition is permanent and irreversible, so be sure you have a backup of the dictionary if the column definitions are critical.

Listing column definitions

When you list the columns for a table, you see these attributes:

Designation	Meaning
Col Name	The name of the column
T	Type: F (field), S (symbolic), G (group)
Pos	For F-type columns, the row position
Mstr	For F-type columns, whether this is a master or synonym.
S/M	Single- or multivalued
Data type	The data type for the column, if one has been assigned
Just	Justification
Len	Display length

To create a report of column definitions for a table:

1. From the development menu, choose **DB Admin-Columns-List.** The List Dictionary window appears.
2. At the *Table name* prompt, enter the name of the table for which you want to list the column definitions. Press [F2] or click <Options> to see a list of available tables.
3. At the *Options* prompt, enter "P" if you want to send the report to your printer.
4. Press [F9] or click <Save> to generate the report.

Creating a symbolic column

Please read "About creating or modifying column definitions", above. Creating a symbolic column is very much like creating a data column, with only these differences:

- Because the column is not associated with a position in the row, there is no position to fill in.
- Instead, you must provide a formula. The formula can use any amount of R/BASIC code up to 34K, but must return a value for the column in the variable @ANS.
- Because no value is entered into a symbolic column, there is no validation pattern. However, all other attributes apply, including data type, output format, display length, etc. These attributes are applied to the value that is calculated in the formula.

To create a symbolic column, follow the steps under "Creating a data column" above, with these differences:

1. At the *Dict type* prompt, enter "S" for symbolic. The *Position* and *Validation* patterns disappear, to be replaced by a *Formula* prompt.
2. At the *Formula* prompt, enter the formula. If you press [F2] or click <Options> at the this prompt, you will enter the Formula window. Press [F2] or click <Options> again to enter the Formula builder. Save your changes in this window by pressing [F9] twice or by clicking <Save> twice.

   For more information about formulas and the formula builder, see the chapter "Building formulas" in the *R/BASIC* book.

If there is a problem in your formula (for example, you've misspelled an R/BASIC keyword or made some other syntax mistake), when you attempt to move out of the *Formula* prompt you will see an error. The error message will tell you (to the best of its ability) what line of the formula contains the error. You must correct the error before you can move out of the *Formula* prompt or save the column definition.

Creating a join column

A special case of creating a symbolic column is creating a column that joins tables together. This type of column uses a formula with the XLATE function. Because this is such a common requirement, however, the formula builder has a special window to help you create the join.

To create a join column, you must:

- Understand thoroughly how tables are related together. See the topics "Related tables" and "Using symbolic columns to join tables" in the chapter "About databases in Advanced Revelation" in this book.
- Make sure that there is a column in the current table that contains (or can derive using a formula) keys to rows in the related table.

To create the join column:

1. Display the Dictionary window (see "Creating a new column", above).
2. At the *Table name* prompt, enter the name of the table in which you are defining the column.
3. At the *Column name* prompt, enter the name of the new column. This can have the same name as the column in the related table, or you can add a prefix (e.g. "CUST_") indicating what the source of the information is.
4. At the *Dict type* prompt, enter "S" for symbolic.
5. At the *Single/Multi* prompt, enter "M" if a) the column to which you are joining is multivalued, or b) if the column in the current table that contains keys to the related table is multivalued.
6. At the *Data type* and *Output format* prompts, enter values appropriate to the data that you will be retrieving from the column in the related table.
7. At the *Formula* prompt, press [F2] or click <Options> to display the Formula window.
8. At the Formula window, press [F2] or click <Options> again to invoke the formula builder. You will see the word "formula" and a popup of starting formulas.
9. Choose the option "@ANS = join". The Join window appears.
10. At the prompt *Name of table to join* enter the name of the related table.
11. At the prompt *Key field used to join tables*, enter the name of the column in the current table that contains keys to the related table. Press [F2] or click <Options> to see a list of columns for the current table.

Caution! Do *not* enter here the name of the column that you are in the process of defining, or Advanced Revelation will produce fatal errors when you use the column in a report or window.

12. At the prompt *Column name of data to retrieve*, enter the name of the column in the related table that you want to display in this table. Press [F2] or click <Options> to see a list of columns in the related table.
13. Save your changes in the Join window with [F9] or by clicking <Save>. You will return to the Formula window, and the word "formula" will be replaced with a formula that uses the XLATE function. Press [F9] or click <Save> to save the formula.
14. Proceed as for any other column. Enter values for justification and display length that are appropriate to the data that you are retrieving from the related table.
15. Save your definition with [F9] or by clicking <Save>.

25. Creating and maintaining indexes

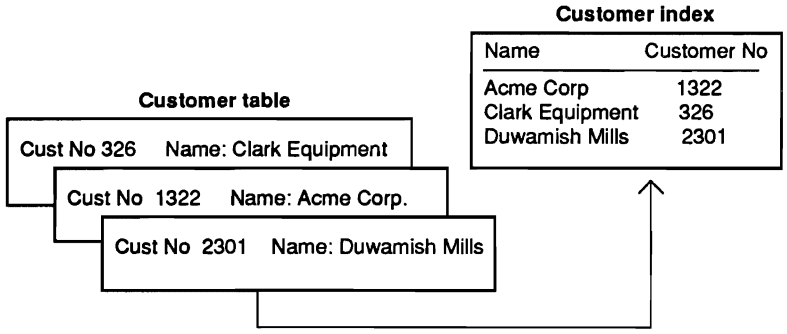
About indexes

Advanced Revelation offers many different techniques to locate rows in your database. The primary access method is for you to provide a key to a row. Advanced Revelation can then use this key to go directly to the row in the table.

If you don't know the key, however, you have to use a secondary way to find it. One way is to search all the way through the table, looking for some data that you know is in the row that you want (for example, an employee name or a particular date). This can be very slow for large tables. The alternative method is to create an index for the table.

Why index a column?

An index is a separate table that stores information in much the same way that an index does in a book. The index contains a list of all the data values in a column (for example, all the company names in a table). Next to each of the data values is the key of the row that contains that value. (In a book index, of course, the "key" is the page number where that information appears.)



The purpose of an index is two-fold:

- An index makes it much faster to find a particular row, since you can look it up by a data value such as a name—without searching individually through each row in the table.
- An index also makes sorting faster because the data values are stored in the index in order. For example, if you index a column containing names, the names are stored in the index in alphabetic order. Because the data is already stored in sorted order, Advanced Revelation doesn't have to sort the data when you want to see it in order.

Which columns should you index?

With such benefits, you might be tempted to put an index on every column in the table. However, consider these trade-offs when you create an index:

- Indexes take up extra disk storage space.
- It takes time to update an index when you've made a change to an indexed column. The more columns are indexed, the longer it takes to update the index.

Therefore, think carefully about what columns to index. To save disk space and to keep indexing efficient, index only the columns that are most often searched and sorted in reports.

For example, in a customer table, perhaps you most often select rows using the STATE column, and usually display the report in company name order. In that case, define indexing for the STATE and COMPANY columns.

You can put an index on any data (F-type) or symbolic (S-type) column. However, there are some rules to follow when putting an index on a symbolic column. These are discussed under "Indexing symbolic columns" below.

About updating indexes

A particularly powerful feature of Advanced Revelation's indexes is that they are updated in the background when your computer is idle. If you are working on a network, other workstations can update your indexes as well. You can even set up a workstation on the network to function as a dedicated index update station.

The strategy of using idle time to update indexes has some implications for how you use indexes in your applications. Therefore, be sure to read the section later in this chapter on updating indexes.

About using indexes

Once you have created an index, there are different ways of taking advantage of it:

- R/LIST, SQL, EasyWriter, and window Query mode use indexes automatically for searching and sorting.
- You can use an index at the key prompt in a window to search quickly for a row in the current table. This is discussed in the chapter "Using windows for queries".
- You can set up an index search as a window prompt option or softkey by using the "B" or "V" codes and commands. Using this facility you can search *any* indexed table while in a window, and to use indexes to find information that is related to the current table. See "Creating windows using Paint" and "Codes and commands" for more details.

- You can access indexes from within an R/BASIC program to search and sort tables rapidly for further processing in your program. See the documentation on the system subroutine BTREE.EXTRACT in the *R/BASIC* book.

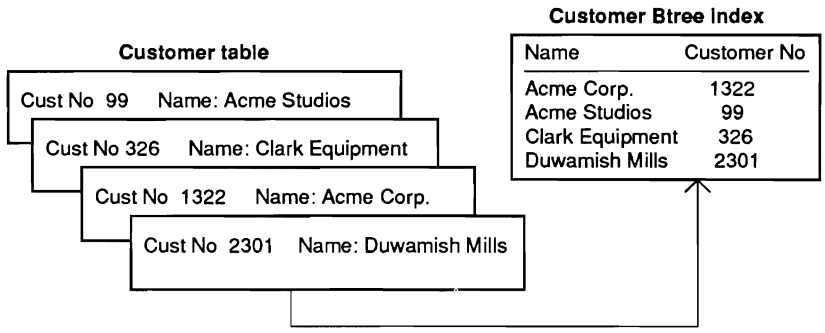
Types of indexes

Advanced Revelation provides four types of indexing: Btree, Cross Reference, Relational, and Quickdex/Rightdex. The kind of indexing that you define depends on how it will be used.

You can specify all types of indexing for a column (except Quickdex or Rightdex, which is used only for row keys). For example, you can specify that a company name column be both Cross Reference and Btree indexed. This section describes the features of the all types of indexes, and when and how to use them.

Btree indexes

Btree indexing uses the contents of the entire column as an index key. For instance, in a Btree indexed company name column, the complete company name would constitute an index entry:



A Btree index stores the entire contents of the column in the index.

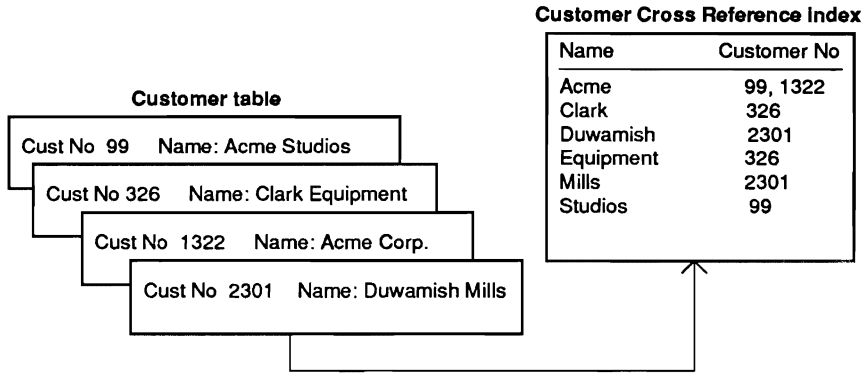
When to use Btree indexes

- When you will typically search the column using the entire contents, such as for ZIP codes, dates, etc.
- When you want to speed up sorting on a column.

You can search on portions of an indexed column (such as part of a name), but this might be slower than using a Cross Reference index (see below).

Cross Reference indexes

A Cross reference index is like a Btree index, but it breaks up the contents of the column into individual words before putting them into the index. In addition, it can throw away words that are so common that they would likely never be used in a search (such as "INC", "THE", etc.):



A Cross Reference index breaks up the contents of a column and indexes each word separately. In addition, it can throw away common words (such as "Corp").

Delimiters to separate words

In Cross Reference indexes you can specify the delimiters (such as space or comma) that define the words to index separately.

Stop list (excluding words from the index)

You can also define the "stop list", the list of words to ignore when searching the index. (Stop list words are ignored in searches, but are still stored in the index). There is a default stop list, and you can define a custom stop list for each index you create.

When to use Cross reference indexes

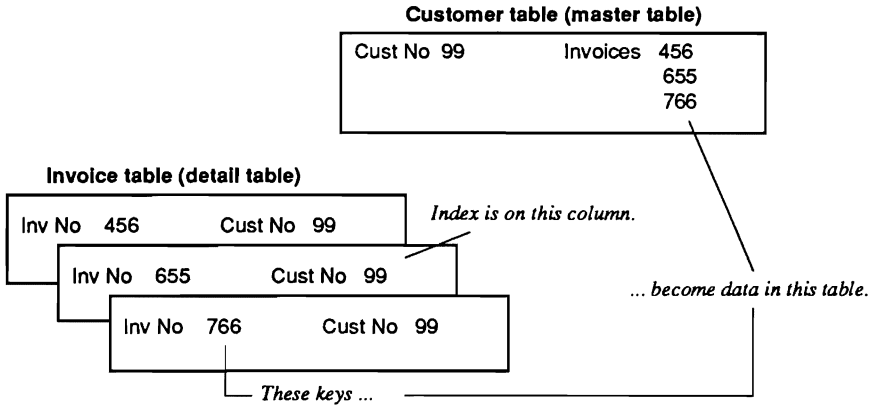
- When you want to index columns that contain many individual words, such as company names, and you intend to search on these individual words.
- When you want to index a column but filter out words, as you might do in a "Notes" or "Comments" column.

Although you can search a Btree index for a word within a column entry, it is faster to use a Cross Reference index for this type of search because the index is maintained as a sorted list of individual words.

Cross Reference indexes don't help when sorting, because each column can have multiple words in the index.

Relational indexes

Relational indexes maintain a relationship between tables. A relational index stores keys from one table as data in another table:



A Relational index stores keys from one table as data in another table. In this example, the source is the Invoice table. The Relational key is the Cust No column in the Invoice table, and the Index column in the destination is the multivalued column Invoices

When to use Relational indexes

- To create a relationship between a master table and a detail table

An invoicing application might make use of a master/detail table relationship. The CUSTOMER table is the master. Each customer master row holds information about a customer who orders goods. The INVOICE table holds detail rows about individual orders. You can have many orders (invoice rows) for a customer. The index is defined for the CUST_NO column in the INVOICE table.

Relational indexes are not used by R/LIST or other search tools, but are used to relate tables together in windows, in the dictionary, and for R/BASIC programs.

Quickdex and Rightdex indexes

A Quickdex index is a sorted index of only the keys in a table. A Rightdex index is the same, but it keeps the keys in right-justified (numeric) order.

Advanced Revelation does not store rows in any particular order in a table. Although you might create rows in numeric order, Advanced Revelation stores them in what seems like random order. However, a Quickdex or Rightdex order keeps a list of the keys in order so that it appears as if the rows were being kept in strict sequence.

You can put a Quickdex on a dictionary as well as on a data table. This speeds things up when you want an alphabetical listing of column names.

Note Quickdex and Rightdex indexes can only store up to 64K of keys. This works out to about 11,000 5-character keys (each key also requires one additional character). If there are too many keys to fit into the Quickdex or Rightdex index, the index is truncated and an error message displays each time the table is updated.

When to use Quickdex or Rightdex indexes

- When you want to keep the rows in the table in order according to their key.
- When you want speed up the display of a list of keys (such as in a popup or dictionary listing).

About indexing symbolic columns

You can use a symbolic column for a Btree, Cross Reference, or Relational index. However, bear these points in mind:

- Indexing on symbolic columns is slower—it takes longer to update the index. Searching the index isn't any slower, though.
- Don't index symbolic columns in which the calculated value is dependent on data from outside the current table. If you do, the outside data might change, and the index will not be updated to reflect the change.

The latter point is particularly important. Consider these two as acceptable and unacceptable examples of indexing a symbolic column:

- **Acceptable.** A symbolic column in an invoice table that calculates the invoice due date based on the original invoice date. This can be indexed safely, because the due date will only change if the invoice date itself changes—a change in the same table.
- **Unacceptable.** A symbolic column in an invoice that derives a customer's company name based on the customer number. This is unacceptable, because if

the company name changes, the change is made in the customer table—not in the invoice table where the index is.

Case sensitivity and indexes

Indexes can be either case-sensitive or case-insensitive. If the index is case-sensitive, the data is stored in the index exactly the way it appears in the row, with uppercase and lowercase letters exactly as they are in the data. A case-insensitive index converts all the data to one form (uppercase) before storing it.

Case-insensitive indexes are the default. The advantage of case-insensitive indexes is that you don't have to search for the exact combination of uppercase and lowercase letters—any combination will work. The disadvantage, however, is that if you *do* want to run a case-sensitive search, the search is slower because Advanced Revelation must go through extra steps to determine if the cases match.

A case-sensitive index is the opposite. It's faster if you often search for data using the exact combination of uppercase and lowercase letters. However, if you run a case-insensitive search, Advanced Revelation ignores the index—so searches run much slower. A further disadvantage is that if you don't explicitly so a case-insensitive search, Advanced Revelation will look only for an exact match on your data, potentially missing entries in the index that are similar but have a different combination of uppercase and lowercase data. For example, if you search for "Smith", a case-sensitive search will miss "SMITH", "smith", and other such combinations.

For information about creating case-sensitive and case-insensitive searches, see the R/LIST section of the *Reference* book.

Creating indexes

You can establish a new index from several locations:

- Using the option **DB Admin-Indexes** from the Development menu.
- Using a the **Tools-Indexing** menu option from within Paint.
- Using the [Shift-F1] softkey in the Dictionary window.

The first two display the same menu and windows, and the last is a simplified version of the indexing options. The steps below focus on the Development menu. However, remember that there are several paths to the same place.

Note If you are going to be making many changes to a table (such as importing into an empty table, or using batch update to change many indexed columns), you will probably find it faster to establish indexes *after* you have made your changes.

Initializing the index

When you create a new index, Advanced Revelation will ask whether you want to update the index right away. If the table you are indexing is large, this can take a long time.

If you will be creating more than one index, you should *not* immediately update the index. Instead, go ahead and define all the indexes that you need. In each case, say "No" when Advanced Revelation asks whether you want to update the index.

When you are all done, you can run a process manually that builds all the new indexes you just created. (See "Rebuilding indexes" later in this chapter for details on this.) Be sure first that you can devote your computer to this task, as it can take a long time, and is a process that can't be interrupted.

Alternatively, if you are on a network, you can let another workstation update the new index for you. See "Updating indexes" below for details on how to distribute index updates around your network.

Creating a Btree index

Before you create a Btree index, you should know:

- The name of the table for which you are creating the index.
- The name of the column that you want to index.
- Whether you want to create a case-insensitive index.

To create the index:

1. Choose the option **DB Admin-Indexes-Btree** from the Development menu. The Btree window displays.
2. At the *Table name* prompt, enter the name of the table to index.
3. At the *Column name* prompt, enter the name of the column to index.
4. At the *Indexing on* prompt, enter "Y".
5. If you want to create a case-sensitive index, change the "N" at the *Case sensitive index* prompt to "Y".
6. Save your changes with [F9] or click on <Save>.

When you save your index definition, Advanced Revelation asks whether you want to update the initial index. If you can wait for the time required (potentially a long time for large tables), answer "Y". Otherwise, indicate "N" and build the index manually later. Note that you cannot search the index until it has been updated.

Note If you are on a network, other users should immediately re-attach the table you have just indexed. If they update the table without re-attaching it, you will have to rebuild the index later to incorporate their changes.

Creating a Cross Reference index

Before you create a Cross Reference index, you should know:

- The name of the table for which you are creating the index.
- The name of the column that you want to index.
- Whether you want to create a case-insensitive index.
- What characters you want to use to separate the individual words in the column.
- What stop list (words to exclude) you want to use. Your choices are:
  - The default system stop list.
  - A custom list that is used only for this index.
  - A combination of the two.

To create the index:

1. Choose the option **DB Admin-Indexes-Cross Ref** from the Development menu. The Cross Reference window displays.
2. At the *Table name* prompt, enter the name of the table to index.
3. At the *Column name* prompt, enter the name of the column to index.
4. At the *Indexing on* prompt, enter "Y".
5. At the *Delimiters* prompt, enter the list of characters that should be used to separate words in the column. Press [F2] or click <Options> to see a multi-select popup of common delimiters.

   The words SPACE, VM (for "value mark"), and TM (for "text mark") are reserved words. However, any other single character you enter will be treated as a delimiter. Always use TM (along with any others you need) when cross referencing a text column that is multiline.
6. At *Stop list modes*, indicate whether you want to use the default system stop list, a custom stop list, or both. Press [F2] or click <Options> to make your choice.
7. If you are using a custom stop list for this indicate, enter the individual words for that stop list at the *Additional stop list* prompt, one word per line.
8. If you want to create a case-sensitive index, change the "N" at the *Case sensitive index* prompt to "Y".
9. Save your changes with [F9] or click on <Save>.

When you save your index definition, Advanced Revelation asks whether you want to update the initial index. If you can wait for the time required (potentially a long time for large tables), answer "Y". Otherwise, indicate "N" and build the index manually later. You cannot search the index until it has been updated, though.

Note If you are on a network, other users should immediately re-attach the table you have just indexed. If they update the table without re-attaching it, you will have to rebuild the index later to incorporate their changes.

When you create a Cross Reference index, Advanced Revelation builds a new column definition called *name*_XREF, where *name* is the name of the column you've indexed.

Changing the system stop list

There is a system stop list for each user and application. When you log onto Advanced Revelation, you use the system stop list that is associated with your user or application.

For more information about users and applications, see the chapter "Creating and managing applications and users".

To change the words in the system stop list:

1. Close the current application or log onto Advanced Revelation with no user or application name. The Opening menu displays.
2. Choose the option **Options-Configuration-Environment-Indexes**. The Index Management Environment window displays.
3. Press [Shift-F1] to display a list of users and applications. Choose the name for which you want to change the system stop list.
4. At the *Words not indexed* prompt, add or delete words as required, one per line.
5. Press [F9] or click on <Save> to save your changes. The modified stop list will be in effect the next time you log onto Advanced Revelation.

Creating a Relational index

Before you create a Relational index, you should be clear about the relationships that you want to maintain. In a master-detail relationship:

- The table being indexed (the source table) is the detail table. The master table is the destination table.
- The column being indexed (the source column, called the Relational key column) is the column in the source table that contains the key to a row in the master table.
- The destination column is always a *multivalued* column in the master table that has been set up specifically to contain a list of keys to rows in the detail table. This is referred to as the Index column.

Once the index is defined, the data in the destination column (the Index column) is protected. You will not be able to make any changes, including deleting that row, unless you remove the index.

In addition to understanding these relationships, you must know:

- Whether you want to create a case-insensitive index.
- How you want to sort the indexed keys. Your choices are:

Code	Meaning
TOP	Put the most recently indexed key at the top (beginning) of the index. In other words, store the keys in reverse chronological order.
BOT	Put the most recently indexed key at the bottom (end) of the index. In other words, store the keys in chronological order.
AL	(ascending, left justified) Put the keys into alphabetical order.
AR	(ascending, right justified) Put the keys into numerical order, lowest to highest.
DL	(descending, left justified) Put the keys into reverse alphabetical order.
DR	(descending, right justified) Put the keys into reverse numerical order, highest to lowest.

The default sort mode is BOT. If you are not concerned about maintaining a sort order for the index, accept the default option of BOT as the sort, because this is fastest.

To create the index:

1. Choose the option **DB Admin-Indexes-Relational** from the Development menu. The Relational window displays.
2. At the *Source table name* prompt, enter the name of the table to index (the detail table).
3. At the *Relational key column* prompt, enter the name of the column to index.
4. At the *Destination table* prompt, enter the name of the table where you want to store the keys (the master table).
5. At the *Index column* prompt, enter the name of the column in the master table where the details keys are to be kept.
6. At the *Sort mode* column, indicate how you want the keys to be sorted in the Index column. Press [F2] or click <Options> for a list of sort modes.
7. At the *Indexing on* prompt, enter "Y".

8. If you want to create a case-sensitive index, change the "N" at the *Case sensitive index* prompt to "Y".
9. Save your changes with [F9] or click on <Save>.

When you save your index definition, Advanced Revelation asks whether you want to update the initial index. If you can wait for the time required (potentially a long time for large tables), answer "Y". Otherwise, indicate "N" and build the index manually later. You cannot use the index until it has been updated, though.

Note If you are on a network, other users should immediately re-attach the table you have just indexed. If they update the table without re-attaching it, you will have to rebuild the index later to incorporate their changes.

Creating a Quickdex or Rightdex index

To put a Quickdex or Rightdex index on a table:

1. From the Development menu, choose the option **DB Admin-Columns-Define**. The Dictionary window displays.
2. At the *Table name* prompt, enter the name of the table onto which you are installing a Quickdex or Rightdex index.
3. If you are putting the index on a dictionary, press [Shift-F4]. If you want to put the index on the data table, press [Shift-F3].
4. When prompted, indicate whether you want to put a Quickdex or Rightdex index on the table. Use Rightdex if the keys are numeric or dates, Quickdex otherwise.
5. The next message asks you whether you want to update the index right away. We recommend that you respond with "Y", as this process rarely takes more than a few minutes. If you choose not to update the index immediately, the index will be built (without prompting) the next time you search the table.

Caution! Do not add or delete rows before updating the Quickdex or Rightdex, as this will result in an inaccurate index. If this occurs, you must remove and re-install the index.

Note If you are on a network, other users should immediately re-attach the table you have just indexed. If they update the table without re-attaching it, you will have to rebuild the index later to incorporate their changes.

Updating indexes

To update an index means to add to it any changes that have been made recently to the indexed table. This is distinct from rebuilding an index, which involves throwing away the current index and building a new one from scratch. For information about rebuilding, see that topic later in the chapter.

Note If you are going to be making many changes to a table (such as importing into an empty table, or using batch update to change many indexed columns), you will probably find it faster to remove the index, then re-index after the changes.

About index updates

In order to be effective, an index must be updated with the latest changes to the table. Because of the way Advanced Revelation maintains indexes, it is critical that you understand how and when to update an index.

The following discussion applies primarily to Btree and Cross Reference indexes. Relational indexes are also affected if you are sharing tables with other users on a network. Quickdex and Rightdex indexes are always updated immediately, so the following doesn't apply to them.

Index update transactions

When you make a change to an indexed column in a table, your change is stored immediately in the data table. However, the index for that column is *not* updated immediately.

Instead, Advanced Revelation create a transaction of the change and stores it in a queue. This takes just a moment, and then the system returns to you, ready for the next action.

The transactions remain in the queue until one of two things happen:

- You leave your workstation idle for a defined period (usually 60 seconds). "Idle" means that you have done no keyboard or table-access activity.
- If you are on a network, another user on the network leaves his or her station idle.

In both cases, the idle machine then begins processing the queue of index transactions, updating the indexes transaction by transaction. If you press any key on the keyboard, Advanced Revelation leaves off updating the index and returns its attention to you.

The advantages of updating index transactions in this way are:

- There is no delay when you update rows in an indexed table—after creating the index transaction, Advanced Revelation returns immediately to you rather than spending time updating the index.
- You put off the potentially time-consuming task of updating the index until your machine is idle.
- You can distribute the task of updating the indexes to other computers on your network, taking advantage of other resources while getting the most performance out of your computer.

You do not at any time lose access to the data in the indexed table. If you know the key to the row, you can recall it immediately, since the primary key isn't dependent on any indexes. You can also search the table on non-indexed columns at any point.

In fact, the only limitation is that you cannot immediately search the *index*. If your computer (or another computer running Advanced Revelation on your network) is idle long enough, of course, the indexes will be updated.

Occasionally, however, you might need to hurry the process and make sure the indexes are current so that you can take advantage of them. Advanced Revelation offers a variety of options for updating indexes all at once rather than waiting for the intermittent idle-time process.

Restrictions on index updates

On a network, only one user at a time can update an index. If more than one user tries to perform index updating at the same time, only the first user will be successful.

Users who attempt to update an index while another user is doing so will be locked out of the index. Advanced Revelation will display a message warning them that there will be a delay before their index update request can be processed. If they choose, they can cancel their request.

Here is a list of the index updating processes that can be run by only one user at a time:

- Creating a new index and then updating it immediately.
- Requesting a manual update or rebuild, whether from a menu, the Command window, or an R/BASIC program.

In addition, while Advanced Revelation's background index processing is performing an index update for one table, no other user can run an index update process on that table.

Note The restriction described above only concerns index *updates*. New indexing transactions can be created at any time.

The update delay message

If one of the processes listed above is already running when a user tries to perform an index update, the following message displays:

```
Another workstation is currently updating indexes.

   The index update you have requested will proceed
   when the other workstation's update is complete.

   You may skip the update you have requested by
   pressing [Esc]. Press any key to continue.
```

When you see this message, you can wait for the current updating process to end. When it does, the message will disappear and your updating process will begin. Alternatively, you can press [Esc] to cancel the update process you wanted to run and try again later.

Updating all indexes

1. Choose the option **DB Admin-Indexes-Update-1** (Update all indexes).
2. Confirm that you want to update all indexes.

Updating all indexes for one table

1. Choose the option **DB Admin-Indexes-Update-2** (Update indexes in a specific table). The Index Table Name window displays.
2. At the *Table name* prompt, enter the name of the indexed table.
3. Press [F9] or click on <Save> to start the update.

Note If you have a very large number of index updates pending, it can be faster to rebuild the entire index. See "Rebuilding all indexes in a specific table" below.

Updating a single index

1. Choose the option **DB Admin-Indexes-Update-3** (Update index for one column). The Index Table Name window displays.
2. At the *Table name* prompt, enter the name of the indexed table. As soon as you enter the name, a single-select popup of the indexed columns for that table appears.
3. Choose the column for which you want to update the index.
4. Press [F9] or click on <Save> to start the update.

Note If you have a very large number of index updates pending, it can be faster to rebuild the entire index. See "Rebuilding all indexes in a specific table" below.

Updating indexes before an R/LIST search

By default, any time you create an R/LIST, EasyWriter, or Query mode search, all affected indexes are updated automatically. You may not want this to happen. For example, you may want to be responsible for index updates yourself so that other users do not get locked out of the index.

You can turn off this automatic update for individual users or for an application. Before you do so, you should understand how to manage the configuration of individual applications and users. For more information, see the chapter "Creating and managing applications and users".

To turn off automatic index updates before R/LIST searches:

1. From the Opening menu, choose the option **Options-Configuration-Environment-Indexes**. The Index Management Environment window displays.
2. Press [Shift-F1] to see a list of applications and users. Select the one for which you want to turn off automatic index updates.
3. At the *Update indexes before filter* prompt, enter "N".
4. Press [F9] or click on <Save> to save your change.

Note The change only affects the user or application you specified. Other users and applications continue to update indexes automatically as before.

Changing the delay before index updates

By default, your computer will wait for 60 seconds of idle time before starting to update indexes. Once it has begun, it waits 30 seconds between each update to see if your computer continues to be idle.

You can change both of these delays for individual workstations. For example, if you don't want your workstation ever to update indexes on idle time, you can set the delay so this never happens.

Note The index delays are set up for workstations (not on a user or application basis). For more information about configuring workstations, see the chapter "Configuring the workstation".

To change the delay before and between idle-time index updates:

1. From the Opening menu, choose **Options-Configuration-Hardware-Workstation**. The Workstation Hardware window displays.
2. At the *Delay before indexing* prompt, enter the number of seconds that the workstation should be idle before starting to update indexes. If you want to turn off idle-time updates altogether, enter zero.
3. At the *Time between index checks* prompt, enter the number of seconds that workstation should wait for keystrokes between index updates. Enter zero if the workstation should do no idle-time index updates.
4. Press [F9] or click on <Save> to save your changes.

Avoiding index update delays

Consider the impact on other users before requesting a an update or rebuild of an index for a large data table. When working with a large table, updating may take a long time. During that time, other users will not be able to perform any process that updates an index in that table, including R/LIST searches involving indexed columns.

You may therefore find it most convenient to limit large-scale index updates or rebuilds to idle time on the network, such as at night or on weekends.

Setting up a dedicated index workstation

If you are on a network, you can dedicate a workstation to doing nothing but index workstations. This frees other computers on the network from this task, while making sure that indexes stay current.

Any workstation on the network can be dedicated to updates. If you have an extra computer (even an older, slower model), you can have a permanent index workstation. But even if you don't want to dedicate a computer to index updates full time, you can turn a computer into an indexing workstation when it would otherwise be idle, such as during a user's lunch hour or day off.

In order to work effectively as an indexing workstation, the station must have access to all the tables for which there are outstanding index updates to be done. This is best accomplished by having that station open the application where the index updates are being made and attach *all* the tables for the application. The index workstation must also be using the same language set and sort order as the stations that are creating the index updates.

Estimating the proper delay

How often should an index workstation check for new index updates? That depends on the activity in the application.

You want to balance the delay between index checks at the index workstation against the number of index updates that other stations are making. For example, if on average there is a new index update every five seconds, then you should set the delay between index checks to 5.

On the other hand, you don't want to set the delay between checks too low, because that generates needless network traffic. If there is a new index update only every minute or so, it's inefficient to have the index workstation check for new updates every five seconds.

Before you set up a dedicated index workstation, you will need to know:

- How to set up a custom workstation configuration, so that you can have a configuration that is used only by index workstations. See the chapter "Configuring the workstation" if you need to know more about this.
- How often, on average, new index updates are being generated in your application.

To establish the index workstation:

1. Create a custom workstation configuration that will only be used by the index workstation.
2. At the workstation that will be used for indexing, log onto Advanced Revelation without an application or user name. The Opening menu displays.
3. Choose the option **Options-Configuration-Hardware-Workstation.** The Workstation Hardware window displays.
4. At the *Delay before indexing* prompt, enter 5. This causes the index workstation to pause five seconds after opening the application before starting to look for index updates.
5. At the *Time between index checks* prompt, enter the number of seconds that you have calculated as the average time between new index updates for the application.
6. Press [F9] or click on <Save> to save your changes.
7. Quit Advanced Revelation and reboot the workstation so that your new configuration is recognized.
8. Log onto Advanced Revelation, opening the application for which you want to dedicate this index workstation.
9. If the startup command for the application doesn't already do so, attach all the tables for which index updates will be created.
10. Leave the workstation keyboard idle. The workstation will begin processing index updates after the period you specified at the *Delay before indexing* prompt.

Stopping a dedicated index workstation

If you want to interrupt an index workstation and revert to using it as a normal workstation, reset the index timeout values to their normal or default settings. Follow steps 2 through 6 above, and fill in the values you normally have for that workstation (the original setting is 60 for *Delay before indexing*, and 30 for *Time between index checks*). Quit Advanced Revelation and re-enter as you normally do.

Removing indexes

Removing an index is generally a matter of following the steps for creating the index, but changing the *Indexing on* prompt to "N". Quickdex and Rightdex indexes are removed by going through the same steps as installing it. You can remove a single index, or all Btree, Cross Reference, and Relational indexes on a table at once.

Removing a Btree, Cross Reference or Relational index

To remove a single index from a table:

1. Display the appropriate index window by choosing **DB Admin-Indexes** from the Development menu, and then the correct type of index from the indexes menu.
2. At the *Table name* prompt, enter the name of the indexed table.
3. At the *Column name* prompt, enter the name of the column for which you want to remove an index.
4. At the *Indexing on* prompt, change the "Y" to "N".
5. Save your changes with [F9] or click on <Save>.

If you remove a Relational index, the index control information is removed. However, the index keys are not removed from the related table, although they cease to be protected from change.

Note If you are on a network, other users should immediately re-attach the table from which you have just removed the index. If they update the table without re-attaching it, they may experience errors and have to log out and then back in.

Removing all indexes from a table

The following removes *all* Btree, Cross Reference, and Relational indexes from a table *except* Quickdex or Rightdex indexes:

1. Choose the option **DB Admin-Indexes-Update-6** (Remove all indexing from a table). The Index Table Name window displays.
2. At the *Table name* prompt, enter the name of the table from which you want to remove indexing.
3. Press [F9] or click <Save> to remove the indexes.

If you remove a Relational index, the index control information is removed. However, the index keys are not removed from the related table, although they cease to be protected from change.

Note If you are on a network, other users should immediately re-attach the table from which you have just removed the index. If they update the table without re-attaching it, they may experience errors and have to log out and then back in.

Removing a Quickdex or Rightdex index

Removing a Quickdex or Rightdex index is essentially the same as installing it:

1. From the Development menu, choose the option **DB Admin-Columns-Define.** The Dictionary window displays.
2. At the *Table name* prompt, enter the name of the table onto which you are installing a Quickdex or Rightdex index.
3. If you are removing the index from a dictionary, press [Shift-F4]. If you want to remove the index from the data table, press [Shift-F3].
4. When prompted, confirm that you want to remove the index.

Note If you are on a network, other users should immediately re-attach the table from which you have just removed the index. If they update the table without re-attaching it, they may experience errors and have to log out and then back in.

Listing indexes

Listing indexed tables and columns

To see a list of all tables currently available that are indexed, choose the option **DB Admin-Indexes-List index.** A popup of all indexed tables appears, with information about the types of indexes (Btree, Cross Reference, or Relational) on each table.

To see more detail about these indexes, highlight a table in the popup and press [Enter] or click on the table with the mouse. A popup of indexed columns for the selected table appears.

To see details about a particular index, select an index from the popup and press [Enter] or click on it.

Printing index reports

You can print the index reports by using the LISTINDEX command at the Command window. Display the Command window by pressing [F5], then enter one of the

following:

LISTINDEX (P)	Print a list of all tables with indexing
LISTINDEX tablename (P)	Print columns with indexing for a table
LISTINDEX tablename columnname (P)	Print indexing detail for a one column

Determining if a table has a Quickdex index

To determine if a table has a Quickdex or Rightdex index:

1. Choose the option **DB Admin-Locations-List** from the Development menu. The Table list window appears.
2. At the *Location* prompt, enter the location where the table resides for which you want information.
3. If you want the report sent to your printer, enter "P" at the *Option* prompt.
4. The resulting report lists all the tables for the location you specified. If the words "QUICKDEX" or "RIGHTDEX" appear in the MFS column next to a table name, then that table has a Quickdex or Rightdex index installed.

Rebuilding (or building) indexes

Rebuilding an index means discarding the current index and re-indexing that column again. Rebuilding is also a way of building an index that you just created. You only need to rebuild under one of these conditions:

- You just installed one or more indexes, and want to create the initial index now.
- Either the data table or index table has experienced physical corruption (a "Group Format Error").
- Users have made changes to a data table after the index was created but before they re-attached the data table.
- The data table has been lost or destroyed by an operating system process.
- You suspect that an index is inaccurate or corrupted and does not contain all the keys.

It can take considerable time to rebuild indexes in a large data table. You may want to run the rebuilding process at night or during a quiet time on the system.

Note If you choose to rebuild all the indexes in one table, the process will be considerably faster than if you rebuild the indexes on individual columns.

Rebuilding a single index

1. From the Development menu, choose the option **DB Admin-Indexes-Update-5** (Rebuild index for one or more columns). The Index Table Name window displays.
2. At the *Table name* prompt, enter the name of the table for which you want to rebuild an index. Press [F2] or click <Options> for a list of indexed tables. A popup of all the indexed columns in that table appears.
3. From the popup, highlight the names of the indexes you want to rebuild, then press [Enter].
4. Press [F9] or click <OK> to start the rebuild process.

Rebuilding all indexes in a specific table

1. Choose the option **DB Admin-Indexes-Update-5** (Rebuild all indexes in a specific table). The Index Table Name window displays.
2. At the *Table name* prompt, enter the name of the indexed table.
3. Press [F9] or click on <Save> to start the rebuild.

Rebuilding a Quickdex or Rightdex index

There is no utility to rebuild this type of index. If you need to rebuild the index, remove it, and then re-install it.

Indexes for international applications

When you create an index, Advanced Revelation stamps information on the index about what language and sort order you are using. During index updates, the system makes sure that the current language and sort order match those of the index. If they do not, Advanced Revelation prevents you from updating the index so that you do not use the wrong sort order, thereby corrupting the index.

For information on how language sets affect indexing, see "How Language Sets Affect Indexes" in the Appendix "International Applications".

Indexing system tables

In general, you should not index Advanced Revelation system tables. At logon, Advanced Revelation attaches some system tables before it attaches the program table

that controls indexing. If you index these tables, Advanced Revelation cannot access the routines is needs to access indexes, and the system freezes.

Caution! If you index system tables, you may have to re-install Advanced Revelation.

Where indexes are stored

When you create the first Btree or Cross Reference index for a table, an index table is created with the name !*name*, where *name* is the name of table you are indexing. All Btree and Cross Reference index key lists for a table are maintained in the !*name* table. For example, the indexes for columns in the CUSTOMER table are held in the !CUSTOMER table.

Caution! Never make manual changes to the !*name* table—always use the index management tools. Any attempt to make changes outside the indexing system can corrupt the indexes a table.

Unlike Btree and Cross Reference indexes, Relational indexes are not stored in the !*name* table. Each Relational index is stored in a column in the master table. Once this column is defined to hold the index row keys, it is protected. It is only updated by the indexing system.

Quickdex and Rightdex indexes are stored directly in the tables. If you put either of these indexes on a table, there is a hidden entry called %RECORDS% in the table that stores the key list. You won't see this row when you list the table, but you can access it if you use the key. Remember that if the Quickdex is on a dictionary, that the hidden entry is in the dictionary, not in the data table.

26. Transaction Processing

About transaction processing

Advanced Revelation offers you a feature called transaction processing for protecting the integrity of your tables. Transaction processing allows you to make complex, multi-file updates with complete assurance that all tables and rows involved in the update are made properly.

In many applications, a single process must update several rows. A simple example is a bank transfer. To transfer money from one account to another, you must debit the first account, and credit the second.

However, something might happen to prevent the tables from being updated properly. One example is a power outage. A less dramatic but more common example is that another user on a network has a row locked that your application needs.

In either case, if only one of the operations took place, the tables would not be synchronized. The result, of course, is a serious error.

Transaction processing prevents this by logically grouping all related updates into a single update operation or "transaction". This transaction is then treated as one update. If any of the individual updates comprising the transaction cannot be completed, the entire transaction is canceled. All the tables that would have been affected by the transaction are left as they were.

In the banking example above, both updates would comprise a single transaction. If either account row could not be updated, the transaction would be canceled. Neither account row would change, and the tables would stay synchronized.

How transaction processing works

Transaction processing stores all the updates you make in temporary tables. Once you start a transaction, the results of any updates you make to tables are assembled in temporary transaction tables rather than being written to the actual data tables.

When you are finished with a transaction, the data is copied from the temporary tables to the data tables. On the other hand, if you abort the transaction, the data is not copied to the data tables, and they remain unaffected.

Any read operations that you include in your transaction will use the results of a preceding update—the row is read from the temporary table. This ensures that you see the most current version of the data. From your perspective, it is as if the changes had actually been made to the data tables.

On a network or in a multi-user system, other users do not have access to your temporary transaction tables. If those users read any rows that you have updated during a transaction, but have not yet written to the data table, they will see an older version of the row.

Applying transaction control

You can use Advanced Revelation transaction processing with any tables in your system. However, unless you are using SQL, you must add transaction control to the table or tables to be monitored.

Note Transaction control is applied automatically to tables being accessed with SQL statements. If you have defined new tables using the SQL CREATE TABLE command, transaction control is already in effect for those tables.

To add transaction control to a table or a group of tables, choose **DB Admin-Table-Transaction processing**. It is not necessary to add transaction control to all tables involved in a transaction. If the updates you are making to a table are not critical to the integrity of a transaction, you do not need to add transaction control to that table.

Note Transaction processing adds slight overhead to the table while monitoring transactions. If performance is a concern, do not add transaction control unless you require it for your application.

For networks or multi-user systems, transaction control also enables you to restrict the maximum number of locks available to a workstation (see "Locking and unlocking" under "Managing transaction processing" below).

Domain controls

When you apply transaction control to a table, you are also installing conversion and validation controls for data. These are called "domain" controls, and have effect only for cursor input and output in R/BASIC.

Note For more information, see the chapter "Programming with R/BASIC" in the *R/BASIC* manual.

Checking for transaction control

To determine if transaction control has already been added to a table, choose the menu option **DB Admin-Volume-List** and at the *Volume name* prompt enter the location where the table is located. If the word "PROTECT.MFS" appears in the *MFS* column for a table, transaction control has been added.

Removing transaction control

You can remove transaction control from a table at any time. To do so, choose **DB Admin-Table-Transaction processing** .

You can remove transaction control from SQL tables with the CONTROL_OFF command; however, when the tables are accessed again using an SQL statement, transaction control will be reapplied.

Using transaction processing

With Advanced Revelation's transaction processing, you have complete control over how and when transactions are processed. You can initiate and complete a transaction using simple commands. Transaction processing consists of these steps:

- Initiating (starting) a transaction.
- Making all the updates required for the transaction. Your changes are stored in temporary transaction tables.
- Committing the transaction to indicate that all updates have been made. This copies the data from the temporary transaction tables to the data tables.

If all goes well, these are the only steps required. However, an additional process is available if you detect a problem before you have committed your transaction:

- Rolling back (canceling) a transaction. This clears the temporary transaction tables.

To control transaction processing, use the TCL command TRANSACTION. This command can be entered at the Command window, or can be executed from within an R/BASIC program with the EXECUTE statement.

Note For more information about the TRANSACTION command, see the "TCL Command Reference" chapter in the *Reference* manual. For more information about programming in Advanced Revelation, see the *R/BASIC* manual.

Starting a transaction

To enable transaction processing, use the TCL command TRANSACTION ON or a call to the R/BASIC TRANSACT subroutine. Once a transaction has been started, all updates are monitored, whether you update tables using a window, an R/BASIC program, or any other process.

Note Remember that transaction processing only monitors tables with transaction control in effect. See "Applying transaction control" above.

There is no predetermined size or duration to your transaction. Any operations you specify from the time you start the transaction until the time you end it are considered to be a part of the same single transaction.

Starting transactions automatically

Using the *Auto-Transaction Start* setting at the Concurrency Control Environment window, you can have transaction processing automatically enabled for all tables with transaction control. This setting is made in an environment window that is specific to a user or application.

To access the Concurrency Control Environment window, choose the option **Options-Configurations-Environment-Concurrency** from the Opening menu. Press [Shift-F1] to choose the user or application for which you want to change this setting.

Automatic processing is then active after you log in or reset the system, and after each commit or rollback operation.

Committing a transaction

When you have finished making all the updates for your transaction, you must close out the transaction. This process is called "committing" your transaction, because you are then committed to the updates you have made.

To commit a transaction, use the statement TRANSACTION COMMIT.

Rolling back a transaction

You may occasionally determine that your transaction is not proceeding properly, or that you made an error during an update. If so, you can cancel the transaction, and roll back the changes made by the transaction.

To cancel a transaction, use the TRANSACTION ROLLBACK statement.

The data tables that were to be updated by your transaction remain in their original state because your updates are never copied from the temporary transaction tables to the data tables.

Pending transactions

If you forget to commit your transaction, when you log off or reset your system, Advanced Revelation will prompt you to commit or rollback the transaction. When you reset you also have the option to ignore the transaction.

After the transaction is committed or rolled back, any locks held by your workstation are released. If you choose to ignore the pending transaction when resetting, the locks are not released and the transaction remains active.

Recovering from a system failure

If your system fails or is shut down after you have already committed a transaction, the updates you have logged to the temporary transaction tables might not have finished copying to the data tables.

If you have not yet committed a transaction when your system fails, all changes you have logged are lost, and the data tables remain in their original, unaltered state. You can only recover transactions that were committed before the system failure.

When you log back into Advanced Revelation after a system failure during the commit process, a message displays telling you that there is an unfinished transaction. You have three choices at this point:

- Finish the commit process that was interrupted.
- Ignore the transaction, and carry on with your work.
- Delete the transaction. This deletes the data in the temporary transaction tables, and effectively rolls back any portion of the transaction that was not completed.

Committing an unfinished transaction

If you choose to commit the transaction, Advanced Revelation will attempt to finish the commit procedure. The transaction process will restart the commit process, copying the data from temporary tables to the data tables.

Ignoring an unfinished transaction

You can have Advanced Revelation ignore the transaction that remains unfinished when you log back in. You might wish to do this, for instance, if you had been updating tables that are not available at the time you log back into Advanced Revelation.

If you choose to ignore an unfinished transaction, you can complete it later using the TRANSACTION RESTART command. This command is a manual version of the procedure that Advanced Revelation uses when it detects an unfinished transaction at logon.

As a rule, you will want to finish a transaction as soon as possible. The changes that you have logged for a table will not be available to you or to other users until the transaction is complete. Other processes or other users may be dependent on changes that you have been logging to a table.

In addition, your unfinished transaction may be holding up transactions that other users have logged. Under some circumstances, transactions are queued for processing. If your transaction is at the top of the queue, other transactions must wait until yours is

completed. Transaction queues are described under "Managing transaction processing" later in this chapter.

Deleting an unfinished transaction

You may decide when you log back on that it is better to simply clear the unfinished transaction. For instance, you may determine that it is more appropriate to finish making table updates manually.

If you choose to delete an unfinished transaction, rows that had not yet been committed when the system failed are removed from the temporary transaction tables. The system also removes control information indicating that you were in the middle of a commit procedure.

Note The delete option does not remove changes that had already been successfully made when the commit process was aborted. If your system failed and the transaction was partially complete, the data tables will be partially updated. Choosing the delete option may therefore result in a loss of data integrity.

Managing transaction processing

The information that follows is intended primarily for advanced users and system administrators. When you use transaction processing, you should be aware of how Advanced Revelation is monitoring and protecting your system. Not only will this help you when you have problems, but it will help you coordinate with other users in a networking or multi-user environment.

Assigning a transaction table volume

The temporary tables used for your transactions need to be stored in a single volume. The default volume is TRANSACT. To assign a different volume for these tables, use the *Transaction table Volume* prompt at the Concurrency Environment window. To display the Concurrency Control Environment window, choose the option **Options-Configurations-Environment-Concurrency** from the Opening menu, then press [Shift-F1] to choose the user or application for which you want to change this setting.

Phases of the commit process

Once you commit your transaction, the rows you have updated are copied from the temporary transaction tables to the actual data tables. As they are written successfully to the data tables, they are removed from the transaction tables.

At this stage, the commit process is only minimally vulnerable to system failure. There are three reasons it is virtually guaranteed to be successful:

1. The updates only begin when you have signaled that the transaction is done. This means that you are confident that the updates you have made are correct and complete. In addition, it is assumed you will not be halting a transaction because of an error that you detect, or because you were unable to access a row you needed.
2. The update of the data table from the transaction table happens in a burst. Because all the updates are concentrated into a very short time, the chances of a system failure during the course of the commit process are reduced.
3. The transaction processing system maintains a log of the updates it makes during a commit process. If the commit process is interrupted (by a workstation or system failure, for example), the log allows Advanced Revelation to restart the update process when you next log on (see "Restarting a transaction" above).

Using the commitlog

When each transaction is committed, user and workstation information is written to a "commitlog". This is a central record of all transactions currently being committed.

All users can display the commitlog. If you are in the SYSPROG application, however, additional functions are available. If you are in the SYSPROG application when you display the commitlog, the commitlog is locked, preventing committed transactions from being executed. This is necessary to give the system administrator an accurate representation of current commits.

If Commit Protection is enabled (see "Queuing committed transactions", below), you are able to remove entries from the queue if necessary.

Assigning a commitlog volume

The table used to maintain the commitlog needs to be stored in a centralized volume. The default volume is TRANSACT. To assign a different volume for these tables, use the *Commit Log Volume* prompt at the Global Environment window. Choose the **Options-Configurations-Environment-Global** from the Opening menu to display the Global Environment window.

Note All users must be accessing the same commitlog volume. If you change the volume for the commitlog, be certain that all users are able to attach this volume.

Displaying the commitlog

To display the commitlog, use the TCL command COMMITLOG. When a commit queue is enabled (Commit Protection is set), the commitlog displays:

- Users with currently committed transactions.
- Workstation identification for each user.

- Data tables having a commit queue.
- Transaction entries in the commit queue for each table.

If a commit queue is not enabled (Commit Protection is not set), the commitlog displays:

- Users with currently committed transactions.
- Workstation identification for each user.

Queuing committed transactions

On networks or multiuser systems, there is a good possibility that more than one user will commit a transaction to a table at the same time. To ensure data integrity, you can choose to protect the orderly execution of committed transactions by enabling queuing. Each transaction is then assigned an entry in a queue for the target table(s) as it is committed.

The *Commit Protection* setting at the Global Environment window enables queuing for transaction updates, and thus assures you of the highest level of data integrity. Choose the **Options-Configurations-Environment-Global** from the Opening menu to display the Global Environment window. The default setting for Commit Protection is "Yes".

Note Serializable transactions are guaranteed only in SQL with a Consistency Level setting of 4 and Commit Protection set. For more information, see the chapter "Managing the system" in SQL, and the Networking section.

When you do not need the measure of data integrity provided by queuing transactions, disabling Commit Protection will improve performance considerably.

Removing users from the commit queue

When committed transactions are queued, each transaction must be executed in order to execute any subsequent transactions in the table queue. For example, if a system failure should occur at a workstation with a transaction in a queue, any remaining transactions in the queue would be unable to execute.

In such emergencies, it may be necessary to remove a transaction entry from the queue. Doing so, however, may risk data integrity by forfeiting the order of execution for commits to the table. A subsequent transaction may access invalid data because of the missing entry.

Only a user in the SYSPROG application can remove a transaction entry from the commit queue. To do so, use the TCL command COMMITLOG to display the current commitlog. When the commitlog is displayed, press [F6] for a popup of the softkeys. There are two operations available:

1. Delete entry from commitlog:

To remove a transaction entry for a specific user from the table queue(s), position the cursor at the appropriate user name and press [Shift-F1].

To remove the first transaction entry in a table queue, position the cursor at the appropriate table name and press [Shift-F1].

This function is disabled if Commit Protection is off.

2. Clear entire commitlog:

   To remove all transactions entries from all table queues, press [Shift-F2].

In both cases, the user's transaction tables are not affected. The entry is simply removed from the commit queue for a table. This enables the user to subsequently restart a transaction, using a TRANSACTION RESTART statement.

Locking and unlocking

If you are working in a network or multiuser environment, generally you will be locking rows before updating them, and unlocking them after the update is finished. During transaction processing, the behavior of these operations is altered slightly in order to guarantee the integrity of the tables.

When you issue a lock during a transaction, the lock is passed through the transaction monitor and set on the network (assuming that the row is available). The subsequent unlock command, however, is not passed through to the network. Instead, it is held by the transaction monitor until you commit your transaction.

This is because, as with any update, the rows in a transaction remain locked until they have been written to a table. Because the changes during a transaction are still in temporary tables, the locks are not released until these changes are written to the data table.

There are two problems you might encounter:

1. You might be keeping other users out of rows for long periods until you commit your transaction, even though within your transaction you are immediately unlocking rows.
2. You might use up a great many locks on your network. Some networks can only maintain a limited number of locks concurrently, and the accumulated locks you set during your transaction may cause the system to exceed this limit.

To counteract these potential problems you can:

- Make your transactions smaller, or perform commits or rollbacks more frequently.
- Eliminate needless locking on non-shareable tables.
- Use table locking where possible, to conserve available locks.

Unlocking all at commit

You can choose to release all locks when you commit a transaction. This includes any locks set by an Advanced Revelation environment or process (such as the Editor or Paint), as well as locks that you might have set yourself.

To release all locks, set *Unlock All at Commit* at the Concurrency Control Environment window. Using this setting will increase performance considerably for large transactions.

Limiting available locks

Applying transaction control also enables a lock limiting function. If you wish to conserve or control the number of locks on your network or multi-user system, you can set the maximum number of locks available to a workstation.

The *Lock Limit* setting is available at the Concurrency Control Environment window. To display the Concurrency Control Environment window, choose the **Options-Configurations-Environment-Concurrency** from the Opening menu. Setting the lock limit to zero removes workstation limits on the number of locks available. This is the default setting.

Note Most networks restrict the total number of locks that can be set. Refer to your network documentation.

Assessing transaction status

You can monitor the status of transaction processing at your station by using the TRANSACTION STATUS statement. This produces a report of the following:

- A list of tables modified during the current transaction.
- The starting date and time for the current transaction.
- An activity index indicating the relative amount of work done during the transaction.

27. Managing volumes

About volumes

While working with Advanced Revelation tables, you will run across references to "volumes". The information in this chapter will explain what volumes are, how to access them, and how to create them.

Basically, a volume is a means of telling Advanced Revelation about the location of a group of tables. At its simplest, a volume is a synonym or alias for a location. However, a volume also incorporates information about the format of the tables on a location. In that case, you can use a volume to tell Advanced Revelation to create or access tables that are on a format other than Advanced Revelation's default format.

Knowing about volumes is useful under these circumstances:

- You would like to use a shorthand or synonymous name for the drive or path name on which your tables reside.
- You are using an Environmental Bond.

Volumes to specify location and type

Whenever you create or attach a table, Advanced Revelation asks you the location of the table. By default, the location is assumed to be an operating system drive or location. Advanced Revelation also assumes that you want to create or attach Advanced Revelation-formatted tables.

In fact, however, these assumptions may not always be correct. Consider these two options:

- You want to create an alias for a long or clumsy path name. In that case, the alias isn't the actual location—it's a synonym for it.
- You are using an Environmental Bond. If that's the case, the data isn't in Advanced Revelation format. In addition, many environments (such as database servers) don't use normal operating system drive or path names to indicate the location of their data.

As an example, you might use the dBASE Environmental Bond to create or access a table in dBASE III format. In that case, you must tell Advanced Revelation both the location *and* that it should create a table in dBASE III format instead of Advanced Revelation, ASCII, or another format.

To indicate both a table type as well as a location, you define a *volume*. A volume is a name that you assign to a group of tables sharing the same table type and location.

For example, all the Advanced Revelation tables on the path C:\AREV\PAYROLL would belong to a single volume. All the dBASE files on the path D:\DBASE\AREVDATA would belong to a different volume, because they are on a different location and in a different format. If there were also Advanced Revelation

tables on the path D:\DBASE\DATA, they would appear on a different volume, because although they are on the same location as the dBASE files, they are in a different format.

Using a volume as an alias for a location

You can use a volume to create a shorthand way of referring to a location. For example, you might be keeping Advanced Revelation tables on the location C:\AREV\WORKING\TEMP. Defining a volume enables you to create an alias for this long path, so that you could use the name TEMP, for example, to create or attach tables there, rather than the full path name.

Using a volume with an Environmental Bond

When you use an Environmental Bond, you define a volume to tell Advanced Revelation what format of data to expect when creating or attaching data. In addition, you add to the volume definition whatever location information is required for that bond to get access to data in a particular place.

Default volumes

Advanced Revelation maintains two default volumes. These have reserved names. They are also attached automatically when you log onto Advanced Revelation.

The system volume (REVBOOT)

Advanced Revelation maintains its system tables (commands, system windows, etc.) in a volume called REVBOOT. The definition of this volume is:

Table type Advanced Revelation

Location Default directory at logon time

The default data volume

Advanced Revelation also uses a default volume for your data tables. If you do not indicate a volume name for certain operations, Advanced Revelation uses the location or volume that you specified as the default for the current application. These operations are:

- Creating a new table (MAKETABLE).
- Attaching a volume or table (ATTACHTABLE).
- Detaching a volume (DETACHTABLE).

- Listing the contents of a volume (LISTVOLUME).

The default data volume uses the Advanced Revelation table type. For information about changing the location and table type of the default data volume, see the topic "Changing the default data volume" later in this chapter.

Volume labels

When you create the first table on a new volume, Advanced Revelation automatically creates a volume directory. Advanced Revelation uses the volume directory to keep track of which tables are available on that volume.

When the volume directory is established, it is assigned a unique label (usually consisting of the date and time). Advanced Revelation uses this label for various internal purposes to distinguish tables on different volumes that might otherwise have the same name (for example, two tables CUSTOMERS on the different volumes C:\AREV\DATA and C:\AREV\ARCHIVE).

If you like, you can establish the directory manually or assign your own label to the volume. You might also have to re-assign a volume label if you copy an entire directory, and then want to access tables on that directory while you have the original directory attached.

Using volumes

Creating a volume

To create a volume, you first assign a unique name to your new volume. You then tell Advanced Revelation the location and table type for the new volume.

Finally, if you will be accessing bonded tables, you might also indicate the location of the "control" tables such as dictionaries, indexes, and the volume directory. For more information about control tables, see the chapter "Introduction to Environmental Bonding".

To define a volume, follow these steps:

1. From the Development menu, choose the options **DB Admin-Location-Define.** The Set Location window appears.
2. At the *Volume Name* prompt, enter a name for the new volume. Use a name that will help you remember something about the volume (such as ASCII_DAT or NETWORK_AREV) You can use any name, as long as:
   - It is unique.

- It is 50 characters or fewer.
- It contains no spaces.
- It contains no punctuation but underscores (_)
- It is not also the name of a subdirectory or drive already available to your system.

3. At the *Table Type* prompt, enter the name of the table type for the volume. Press [F2] or click <Options> to see a list of possible table types. If you are creating a volume that is an alias (but otherwise uses Advanced Revelation tables), be sure to choose the Advanced Revelation table type.

   Advanced Revelation fills in information about the table type, either "RTP57" for the Advanced Revelation table type, or a name with the suffix "_BFS" for Environmental Bonds.

4. At the *Data Location* prompt, enter the path or drive name for the volume. If you are using a remote storage location such as a database server or a larger computer, enter the name by which that location is recognized from your operating system. Some bonds provide a window at this point to help you define the location. If so, you can press [F2] or click <Options> to access this window.

5. If you are not using an Environmental Bond, skip the *Control Tables Location* prompt.

   If you are using an Environmental Bond such as ASCII or dBASE, fill in the path or drive name where you want to store Advanced Revelation system control tables (dictionaries and indexes) that the bond requires. By entering a path at this prompt, you can keep your Advanced Revelation control tables separate from the data tables in your bond.

6. Press [F9] or click <Save> to save the volume information.

Attaching a volume

Once you have created a volume, you can use its name at any prompt in Advanced Revelation that asks for a location. For example, to attach the tables on a volume, use the Attach window, and at the *Location or volume* prompt, enter the name of the volume that describes the combination of location and table type that you want to attach.

Displaying tables on a volume

To see a list of all the tables on a volume, choose the options **DB Admin-Location-List**. The ListVolume window displays. At the *Location or volume* prompt, enter the name of the volume for which you want to see a table listing. Press [Shift-F2] at the

prompt to see a list of all volumes. The volume does not have to be attached in order for you to list the tables.

Changing the default data volume

Advanced Revelation uses the default data volume if you do not indicate a volume when you create or list tables in a volume. Each application you create has its own default data volume. The table type for the volume is Advanced Revelation.

You can change the location or table type for the volume or both. This allows you to use a different subdirectory or drive as the default location for your tables.

A default data volume is established as an environment setting. For more information about environments, see the chapter "Configuring the workstation".

To change the default data volume, follow these steps:

1. Create a new volume following the steps listed under "Creating a volume" earlier in this chapter. When you do, enter your choice for the location and table type that you wish to use as the default.
2. From the Opening menu, choose the option **Options-Configuration-Environment-General**. The General System Environment window displays.
3. Press [Shift-F1] to display a list of applications. Choose the application for which you are changing the default data location.
4. At the prompt *Default data volume*, enter the name of the volume you have just created. Save your change with [F9] or by clicking <Save>.

The next time you open that application, Advanced Revelation will automatically attach the volume you have entered as the new default. In addition, if you create a new table without indicating a volume, it will be on the new default volume.

Note If you have tables on the old default data volume, you will need to attach them manually to access them. Advanced Revelation will no longer automatically attach the tables.

Establishing or labeling the volume directory

You can establish or change the label on a volume. If you assign a volume label yourself, you must ensure that the label is unique among all volumes in your system. This guarantees that processes such as indexing will work properly.

Note You cannot change the label of a volume that has indexed tables on it.

To establish or label a volume directory, you must use the TCL NAMEVOLUME command. Follow these steps:

1. Press [F5] to display the Command window.
2. Enter this command:

   ```
   NAMEVOLUME volumename label
   ```

   where *volumename* is the name of the volume whose label you want to choose, and *label* is the new label. The new label can contain no spaces or quotes.

Using Advanced Revelation with Networks

28. Introduction to Networks

Introduction

Advanced Revelation is an ideal environment for use on Local Area Networks (LANs). Whether you are designing new applications or simply using existing ones, you will find that Advanced Revelation's efficient design and powerful tools help your network become the productivity tool it was meant to be.

In this chapter, you will find general information about how to use Advanced Revelation with your network. Subsequent chapters in this section describe how to manage your Advanced Revelation system on a network and provide details about such important issues as table and row locking.If you work on a Local Area Network (LAN) — or if you plan to do so in the future — you will find that Advanced Revelation is an ideal application development environment. Advanced Revelation's powerful application development tools and flexible architecture allow you to build applications easily while still taking full advantage of the capabilities of the network.

In using Advanced Revelation on a network, you will enjoy these benefits:

- Advanced Revelation's efficient filing system minimizes network "traffic". Advanced Revelation is particularly efficient for high-volume, transaction-based applications.
- Response over the network remains the same even as tables grow larger.
- You can store Advanced Revelation data across any combination of local and network media, including multiple servers and remote devices.
- Advanced Revelation offers a full array of options for coordinating data access with other users ("locking"). You can lock tables or rows, and you can have Advanced Revelation lock automatically or take full control of over how and when locks are set.
- You can even use your network for distributed processing. Advanced Revelation's indexing system allows you to share the task of indexing across multiple work-stations. In addition, you can use database servers for optimizing data access, especially for large-volume processes and reports.

About networks

A network is a group of computers or "workstations" that are linked together to share resources and data. Typically, there are two resources that network work- stations share: disk storage and printers. More sophisticated networks may also offer the opportunity to share a resource such as a communication link to another network or larger computer.

Networks can be simple or complex. A simple network, for instance, might link two PCs together using a length of wire similar to that used for cable television. Large networks, or WANs (Wide Area Networks), on the other hand, might actually span

huge areas from buildings to entire states, linking workstations through cabling, telephones, infrared and microwave channels, and more.

Advantages of a network

The first benefit of a network is cost. By sharing resources, you save the cost of equipping each PC in your company with its own hard disk and printer. Instead, each user on the network can share disks and printers, treating them as if they were local to the station.

A far more important benefit than sharing *hardware*, though, is sharing *information*. By tying in to a network, users can access a single, central database of information.

For example, you can maintain a central database of customers and invoices on a network. By having each user on the network use that central database, you avoid having separate copies of the data for each user. Not only does having a central database save time and space, but by maintaining just one copy of the data, you ensure the integrity of the database. With only one copy, you do not need to worry about which copy of the data is current.

Types of networks

There are many types of networks available today. Features and functions vary from one brand to the next, but the basic function of all networks is the same.

Because of the way Advanced Revelation communicates with your network, you rarely have to concern yourself with network differences—in fact, it is only when you configure Advanced Revelation for your network. Advanced Revelation is designed to minimize the need for user interaction with the network.

Network Hardware

Network hardware includes everything required to physically link computers together. This usually consists of a card that is inserted into a PC, plus cabling to connect it to other PCs and to the central PC.

Your choice of network hardware affects the cost and performance of your network. However, as a rule, the hardware has little to do with how you run software such as Advanced Revelation on your network.

Network Software

Far more important to your work with Advanced Revelation is the network software that you use. The network software, or network operating system (NOS), does for your network what your PC operating system (DOS or OS/2) does for your PC: it coordinates activity on the network and provides uniform, high-level access to the hardware.

For example, your network software will determine how you can access printers and hard disks on your network. The network operating system also determines such factors in your network as security and data coordination (locking).

Note Whenever you see reference in Advanced Revelation to a "network type" or simply "network", it *always* means network *software*. For example, if you see reference to an "IBM" or "Novell" network, this refers to the brand of network operating system software that you are using. Do not confuse network hardware and software!

Servers and workstations

Most networks make a distinction between workstations (usually PCs) and servers. A workstation is the computer that you use for your own processing. For instance, when you use Advanced Revelation on a network, you are actually executing Advanced Revelation commands and processes at your local station (even though they are stored on the network).

When tied into a network, workstations use the resources of a "server". The server is a computer dedicated to "serving" requests from workstations. Most requests take the form of a request for data (using the server's disks) or for printing (using a printer attached to the server).

The type of server just described is a "file server". A file server accepts requests for tables and usually provides a way for network users to print to a central printer. The table server does little or no processing of the data — any processing tasks are left to the workstation itself.

A different type of server is a "database server". This type of server is specially designed to accept not just requests for tables, but requests for specific rows. For example, a database server will allow you to send a request such as this: "please retrieve all rows for customers with open invoices, sorted according to zip code". The database server actually examines the data and returns only the rows that meet your request.

Advanced Revelation and networks

In many ways, using Advanced Revelation on a network is no different from using it on a stand-alone PC. Certainly much of what you do with Advanced Revelation is equally applicable in a single-user or a network environment: building and using windows, maintaining your database, running reports, and so forth.

However, there are important differences. For one thing, you must "bump" your single-user copy of Advanced Revelation before you can use it on a network. In addition, you must coordinate data access with other users (set "locks"). The rest of this chapter addresses general issues that arise when you use Advanced Revelation on a network. For additional information on using Advanced Revelation on a network, see the chapters "Managing Advanced Revelation on a network" and "Locking", which follow.

Network compatibility

You can use Advanced Revelation on all major networks. There are network "drivers" available for the best-known networks (see "Preparing Advanced Revelation for a network" below).

If you use a network that does not appear on the list of supported networks, you will probably still be able to use Advanced Revelation. Even lesser-known networks generally support or emulate one of the networks supported by Advanced Revelation. In addition, Advanced Revelation supports a "generic" network that uses the very popular Microsoft MS-DOS 3.1 protocol.

For example, a small network may use the protocol of a better-known system such as IBM or Novell. As long as the emulation is complete, Advanced Revelation will work fine because it can issue commands as if communicating with the better-known system.

Note As a rule, Advanced Revelation requires that you use a "dedicated" file server. This means that you cannot use the server as both a server *and* a workstation. Even if your network allows the use of non-dedicated servers, you may not be able to use Advanced Revelation in this configuration. Consult your network documentation and the LAN Pack documentation (see below) for details on using non-dedicated servers.

Preparing Advanced Revelation for a network ("Bumping")

When you buy a copy of Advanced Revelation, you buy a single-user system. To use Advanced Revelation on a network or for multi-tasking, you must "bump" it with one or more LAN Pack disks. LAN Packs are sold separately.

Caution! *Never* attempt to share data with other users on a network unless you have bumped your copy of Advanced Revelation. If you do not bump first, data corruption is inevitable.

Bumping a copy of Advanced Revelation allows you to:

- Support multiple users or multiple tasks.
- Specify the type of network that you are running on.

The first time you bump your copy of Advanced Revelation, it turns from a single-user system into a system that supports up to five users at a time. If more than five users need to use Advanced Revelation concurrently, you can bump again with a new LAN Pack disk. Each bump adds five more users onto your copy of Advanced Revelation. For example, to support a total of 20 users, you need to bump four times using four LAN Pack disks.

When you bump, you also specify the type of network that you are running on. This enables Advanced Revelation to install the appropriate network driver for your system. Although this is part of the bump process, you can change network drivers at any time.

Note Instructions on bumping your system and choosing a network driver are provided with the LAN Pack.

For information on using multiple copies of Advanced Revelation on a network, see "Sharing data among copies of Advanced Revelation" in the chapter "Locking". Also see your LAN Pack documentation.

Coordinating access to data

Because multiple users on the network have access to the same data, more than one person might attempt to update a row at the same time. This access conflict could result in either:

- one user overwriting another's changes to a row without either one knowing, or
- the table being corrupted and data lost.

To avoid conflicts of this sort, Advanced Revelation allows you to "lock" rows and tables as you use them. By locking a table or row, you are saying "this table (or row) is in use". Other users who attempt to lock the same table or row are not allowed to do so.

You can update a table or row while you have it locked. Since you are the only user who has the table or row locked, you are assured that no one else will be updating it at the same time. When you are through updating, you unlock the table or row. Other users can then lock and update the table or row.

Advanced Revelation offers you different levels of locking tables and rows. You can lock an entire table or just an individual row. In addition, you can choose different degrees of locking, including "exclusive", "shared", and "coordinated" locks.

Note Information about locking and lock types appears in the chapter "Locking" in this section.

Writing network applications

Advanced Revelation allows you to write applications that can be used in both single-user and network environments. For example, if you are working on a single PC, but anticipate that your application will be run on a network, you can build *all locking into* your application as you develop it.

Most tools and utilities that you use in Advanced Revelation are network-ready. For example, the Advanced Revelation editor does row locking as you edit rows. Advanced Revelation windows even allow you to specify the type of locking that you wish to use

by simply entering your choice at a prompt in Paint. If you use R/BASIC, Advanced Revelation's programming language, you take complete control over how and when locking is done.

If you run in a single-user environment, locks are ignored. However, when you move to a network, the locks built into your application are passed through to the network, assuring you of coordination with other users.

About network users

Advanced Revelation makes a distinction between users and workstations. A user is anyone who logs into Advanced Revelation over a network. This is different from workstations physically attached to the network.

For example, in Advanced Revelation terminology "five users" means that there are five users logged into Advanced Revelation at the same time. These five users can be logged onto any five workstations on the network.

Note If you are working with OS/2, you are still considered only one user, even though you can run multiple sessions from your workstation. For details on other multi-session systems, see the documentation that accompanies your network software and the Advanced Revelation optional module for that system.

About station numbers

Each network workstation has a unique identification for Advanced Revelation. This "station number" is usually derived directly from the network, using a logical or physical address. The station number is important in a network environment because it is used to uniquely identify processes, tables, and rows.

Note You should be sure that every workstation on your network can return a unique identification.

If you are an R/BASIC programmer, the station number is available to you in the system variable @STATION.

Displaying network information

If you need to see information about your Advanced Revelation network configuration, choose the option **Help-System Info** from the Opening or Development window, or use the TXL command WHO at the Command window. The WHO window displays the number of network users to which you have bumped, the type of network driver that you have installed, and your station number.

29. Managing Advanced Revelation on a network

Managing tables on a network

Under most circumstances, using tables on a network is no different than using tables on a stand-alone system. However, because you will be sharing data with other users, certain issues arise that do not apply to a single-user system.

Accessing tables

When using a network, you can access any combination of local and network tables concurrently. For example, you may be accessing shared customer and invoice tables on the network, while maintaining a local table for your own to-do list or contact information.

If you do access local tables, remember that others on the network will probably not be able to read or update these tables. Therefore be careful about storing data locally if you wish others to have access to it.

Although you can attach any combination of network and local tables, you should only access network tables if you are logged onto a network version of Advanced Revelation. See "Logging on to Advanced Revelation" elsewhere in this chapter for details.

To access tables on a network, simply attach them as you would a local table. You can use the Attach window or the TCL command ATTACHTABLE, or you can create a volume that includes the network location of your tables.

Note For information creating volumes, see the chapter "Managing volumes".

When indicating the location of the table, use the path name for the table as understood by your network. For example, if your network has available a logical drive such as "T:" you can simply use that drive location when you attach the table or create a volume. If you are accessing data from multiple servers, you might use a path name something like this:

```
ATTACH ADMIN\SYS:PAYROLL
```

Creating and deleting tables

You can create a new table on a network any time, just as you can on a stand-alone system. However, before other users on the network can access the new table, they must attach it.

Therefore, when you create a new table that others will share, you should alert them. If they need to use the new table, they can then attach it. Use the facilities of your network to tell others about the new table, including its location.

The same is applicable when you delete a table. As soon as you have deleted a table, alert other users that they should detach the table. If they continue to attempt access to a table that has been deleted, they will experience errors.

Table types

If you intend to share data with other users, you *must* use a table type that can support table and row locking. In Advanced Revelation, the only table type that does this is Linear Hash (the default table type). Therefore, always choose the Linear Hash table type when defining and creating tables for shared access.

Caution! Do not attempt to share data with other users if you are not using a table type that supports locking. If you do, you will experience logical or physical table corruption.

If you are using an Environmental Bond, the table you are using may allow you to share data with other users. For example, the dBASE bond supports locking, so that you can safely share the table with other users on a network. For information on whether your Environmental Bond supports locking, see the documentation that accompanies your bond.

To safeguard data in tables that do not support locking, you may wish to use your network operating system to set the tables' status to "non-shareable". See your network documentation for details. Linear Hash tables must *always* be set to shareable using your network operating system, even if you do not intend to share data with other users.

The Advanced Revelation environment

Because multiple users will be accessing the same copy of Advanced Revelation, it is critical that all users understand how to access a network copy of Advanced Revelation properly. In addition, it is more important to use the built-in facilities for creating unique users and environments.

Logging on to Advanced Revelation

If you intend to access Advanced Revelation tables on a network, you should be using a network version of Advanced Revelation. A network version of Advanced Revelation:

- Has been bumped to support four or more users.
- Resides on a network drive.
- Is the same copy of Advanced Revelation that others on the network are using.

You should not attempt to access Advanced Revelation tables on a network drive unless your copy of Advanced Revelation has been bumped. If you use a single-user copy of Advanced Revelation to access tables, you cannot lock tables or rows. As a result, your access to tables can conflict with that of other users, and the result might be data corruption.

You should also be sure that you are using a copy of Advanced Revelation (the AREV.EXE file) that is on the network. In general, you should log on to Advanced Revelation using this sequence:

1. Log on to your network.
2. Switch to the drive or subdirectory where you store your network copy of Advanced Revelation.
3. Log on to Advanced Revelation.
4. Attach tables that your application requires.

Do *not* use a path name or search drive when you log onto Advanced Revelation. For example, you cannot be on drive T: and invoke Advanced Revelation using the command F:\AREV.

Make sure that you are using the same copy of Advanced Revelation as everyone else. In other words, do not keep multiple copies of Advanced Revelation on your network. If you and other users are using separate copies of Advanced Revelation, you may not be able to lock tables and rows correctly, and may experience data corruption.

Note Under certain circumstances, you can use multiple copies of Advanced Revelation on a network. For details, see "Sharing data among copies of Advanced Revelation" in the chapter "Locking" in this section. Also see your LAN Pack documentation.

Finally, you may find that some networks use quite a bit of memory in your PC when you are logged on to them. Be sure that you have at least 512K of RAM available when you attempt to log onto Advanced Revelation. If there is not sufficient memory, you may need to reconfigure your network software (such as your network device drivers) so that you can free up sufficient RAM for Advanced Revelation.

Users and environments

Environment settings

When using Advanced Revelation on a nework, you can change environment settings for individual users or workstations. Particular settings that you may wish to customize include:

- *Rollout file.* If you use the menu option **Exit-DOS** or use the SUSPEND command, Advanced Revelation stores information in an operating system file

until you use the EXIT command to return. See "SUSPEND" under "Executing network commands" later in this chapter.

Rollout file names are workstation-specific. To change the rollout file for a particular workstation, from the Opening menu choose **Options-Configuration-Hardware-Workstation**, and change the path and file name at the *Rollout DOS file name* prompt. For more information about workstations, see the chapter "Configuring the workstation".

- *Transaction table location.* You can establish a separate location for each user's transaction tables. From the Opening menu, choose **Options-Configuration-Environment-Concurrency** and change the setting at the *Transaction table location* prompt. For more information see "Transaction processing on a network" later in this chapter.

Global environment settings

A small set of environmental settings must be the same for all Advanced Revelation users on a network. These global settings cannot be customized for an individual user on the network.

Global environment settings pertain to transaction processing, to implicit locking in SQL, and to coordinated locking. For information on changing these global environment settings, see the chapter "Transaction processing", and the chapter "Locking" later in this section.

If you change a global environment setting, *all* users must log off and log back on in order to read these settings.

Executing network commands

You can run network commands from within Advanced Revelation just as you can run operating system commands such as DIR, COPY, and FORMAT. Use the menu option **Exit-DOS** or the TCL PC or SUSPEND commands to temporarily leave Advanced Revelation. To return from the operating system to Advanced Revelation, use the EXIT command.

Caution! Do not change or set new network drive mappings for Advanced Revelation tables while you are temporarily suspended from the system. If you execute PC or SUSPEND and then change your drive mapping, the system will not lock data until you log out of Advanced Revelation and log back in.

Using the network command processor

As with running operating system commands, your computer must be able to find and load the command processor for your operating system. For example, in DOS this is established using the command similar to this:

```
SET COMSPEC=C:\COMMAND.COM
```

It is particularly important when on a network that you use the version of the command processor (COMMAND.COM) that is correct for your computer.

Note For information on using the command processor from within applications, see your network documentation.

SUSPEND

If you use the SUSPEND command, Advanced Revelation is stored in an operating system file. You can establish the path and file name for this file using the TCL SETROLLOUT command or a workstation setting for the rollout file. To use SETROLLOUT, see the "TCL Command Reference" chapter in the *Reference* manual. For information about workstation settings, see the chapter "Configuring the workstation".

Note The OS/2 operating system does not need to store Advanced Revelation in an operating system file. If you are using Advanced Revelation for OS/2, SETROLLOUT has no effect. SUSPEND is identical to the TCL command PC.

Printing on a network

You can print easily to a network printer from within Advanced Revelation. Most networks allow you to redirect your printer output to a printer that is attached to the network. Printing to the network from within Advanced Revelation is simple:

1. Redirect printer output to the network printer.
2. Generate your printer output as you would for a local printer.
3. Indicate to the network that you are finished printing. On most networks, the printer only begins to print after you execute this step.

In order to accomplish steps 1 and 3, you must be able to execute network-level commands. See "Using the network command processor" above for details.

This example (for a Novell NetWare network) shows how you might print to a network printer:

```
PC EXIT CAPTURE /P0
LIST CUSTOMERS BY CITY NAME ADDRESS CITY (P)
PC EXIT ENDCAP
```

In the example, the first command redirects output to network printer zero (P0). The second command generates printer output, and the third command indicates to the network that you are finished using the printer.

Note For more information on printing from Advanced Revelation, see the chapter "Configuring the workstation".

Other network management issues

In addition to the general topics described earlier, there are network issues that pertain to specific areas of Advanced Revelation. These include SQL, Transaction Processing and R/BASIC programming.

Implicit locking

Advanced Revelation SQL uses "implicit" or automatic locking to control access to tables by multiple users. Because much of the decision-making for multi-user coordination is done by SQL itself, there are several environment settings that apply to implicit locking. These are described fully in the chapter "Locking".

Transaction processing on a network

Transaction processing on a network is largely the same as it is on a stand-alone system. However, because multiple users will be using transaction tables, you must consider several factors.

Note For more information about transaction processing, see the chapter "Transaction processing".

Transaction Location

First, each user can determine where they wish to store temporary transaction tables. These tables are used to hold updates until the user commits the transaction.

Temporary transaction tables can be located anywhere (local or network), and can be in a different location for every user. Locating transaction tables on a local drive can reduce network traffic. To establish a location for transaction tables, use the Concurrency environment window (**Options-Configuration-Environment-Concurrency** from the Opening menu).

Central Commit Log

While a transaction is being committed, it is logged in a central commitlog. *The location on which this log is maintained must be the same for all users on a network.* Because of this, you set the location of the commitlog using the Global environment window (**Options-Configuration-Environment-Global** from the Opening menu).

Note If you are using more than one Advanced Revelation system on your network, be sure that all copies use the same location for the central commit log.

Commit Protection

On a network, multiple users can be committing transactions at the same time. This may mean that several users are committing to the same table at the same time.

By default only one user can be committing against a table at a time. In order to accomplish this, separate queues are maintained for each table involved in a commit process. When a transaction attempts to commit (or restart a commit), entries are placed in the queue for each table updated during the commit. The result is that tables are updated chronologically, according to the time when a transaction was committed.

By setting the *Commit Protection* prompt in the Global environment to ON, you specify that you want this one-at-a-time protection. Setting this prompt to OFF boosts commit performance. However, you may sacrifice data integrity, because multiple users can commit at the same time to the same tables. There is no guarantee of chronological updates to tables.

The COMMITLOG Utility

You can view and maintain the commitlog and associated table queues using the TCL command COMMITLOG. This may be necessary, for instance, if a workstation experienced a failure while in the middle of a commit process. If commit protection is on, other users may be prevented from finishing their commits. The COMMITLOG utility allows you to clear the first user from the commitlog, so other users can proceed.

A user in the middle of a commit operation may already have updated data tables. If you clear a user from the commitlog, the updates already made by that station are not backed out. As a result, there may be logical data corruption in the tables. You will need to manually finish or back out changes made by that user.

R/BASIC programming for network applications

The applications that you write in Advanced Revelation might require programs written in R/BASIC. If so, you must take extra steps in order to ensure network compatibility for your programs.

Note You can write all your programs with the idea that they will be run on a network version of Advanced Revelation. If they are run on a stand-alone PC, the network logic will be ignored.

Locking

If your programs will be updating table or rows, they should set locks before reading rows. You must use the R/BASIC LOCK and UNLOCK statements in your programs — there is no other mechanism in R/BASIC for coordinating table access with other users.

See the chapter "Programming with R/BASIC" and the commands LOCK and UNLOCK in the chapter "R/BASIC Command Reference", both in *R/BASIC*, for more information.

The system variable @STATION contains the station identifier for your workstation. You can use this number to tag tables, rows, or processes uniquely on a network. For example, if multiple users might be writing log entries to a table, you can use @STATION as part of the log entry to identify which station it came from.

30. Locking

Introduction to locking

One of the major requirements of a network-ready system is the ability to coordinate the requests of multiple users. To do this, Advanced Revelation allows you to "lock" tables and rows.

Purpose of locking

Whenever you are accessing data on a network, there is always the possibility that another user may also wish to access the same data. In order to coordinate multi-user access like this, Advanced Revelation offers a variety of table and row locking methods.

When you lock a table or a row, you are indicating that no one else should attempt to update it. In effect, it is like posting an "in use" sign on the table or row.

While you have the table or row locked, you can update it. When you are done, you unlock the table or row. This clears the lock you have set, and other users may then perform updates.

Locks vs. data access

It is important to understand that locking a table or row is separate from accessing it. When you lock a row or table, you are not really physically preventing another user from accessing it. Instead, you are preventing that user from *locking* it. *Other users can still access data while you have it locked.*

This means that in order to coordinate data access, all users must use "locking protocol". If another user wishes to update a table or row, that user must always first lock the table or row before doing so.

Locking done this way is sometimes referred to as "logical" locking, since there is no physical lock mechanism — only a "semaphore" that is interpreted as a lock. As an analogy, compare how traffic lights work. A red light means "stop", but it does not physically impede you from going through an intersection. However, in order to prevent accidents, drivers adhere to the "traffic light protocol", stopping on red and proceeding on green.

It is an advantage that you can still access a locked table or row. For example, suppose that one user has locked a parts table preparatory to applying global price changes. If the table is large, it may remain locked for a long time while the price changes are being made.

Other users, however, may need to look up the names of parts (knowing that the prices are not accurate). Because they do not intend to update the parts row, they might read the row *without* locking it. They are not attempting to lock the row, so their access

does not conflict with the table lock established by the user who is updating the prices. If they *did* attempt to lock parts rows, they would be refused.

Locking your own locks

A feature of Advanced Revelation locking is that it can detect if you have already set a lock on a table or row. For example, if you are using the Advanced Revelation editor, you might call the Command window, and then (perhaps inadvertently) attempt to edit the same row again.

If you already have a table or row locked, a subsequent attempt to lock it again will fail, just as if another user set the lock. This preserves data integrity, since you might otherwise be making different changes from two or more places in the system.

If you are an R/BASIC programmer, you can determine that you have already locked a table or row. When a lock attempt fails, you can test the status of the lock to determine if you or another user has set the existing lock, and proceed accordingly.

Locking in a single-user environment

Whether you are running a network or a single-user version of Advanced Revelation, you can always use table and row locking. In a network environment, your locks will be passed through to the network. In a single-user environment, your locks will be ignored, and there will be no effect on your applications.

Types of locking

When you use locking in Advanced Revelation to coordinate access to data, there are many options. These involve:

- At what level to lock (a whole table or just a row)
- What degree of locking you need (exclusive or shared)

As you use different facilities within Advanced Revelation, you will have the opportunity to choose from among these options. (Some processes do not give you a choice — they are pre-defined to use a particular type of locking.)

Note For more information on when and where you can choose locking options, see "Establishing locking" later in this chapter.

Levels of locks

Your first option is in determining at what level to set a lock. You can lock an entire table at once, giving you full control of the table. Alternately, you can lock a single row at a time, and not tie up tables for other users.

Table Locking

When you have a table locked, other users cannot also lock that table. Use table locking when you need to:

- Make extensive changes to a table.
- Conserve the number of locks used.

Since some networks limit the number of locks you can set, it may be impractical for you to lock each row you update individually. If you will need to lock a great many rows concurrently (before unlocking them), it is more effective for you to use a table lock.

Note Other users *may* be able to lock individual rows within the table (see "Coordinated locks" below for details).

Row Locking

When you have a row locked, other users cannot lock the row you have locked, but they can lock other rows in the table.

Use row locking when you need to:

- Make specific changes to a table.
- Allow multiple user access to a table.

Degrees of locks

Whether you are locking a table or a row, you can set different *degrees* of locks. By specifying a degree of locking, you can share access to the data, or restrict all other users.

Note Your network may not support all of the methods described below. Refer to your network documentation for information.

The locking methods operate according to an established priority. You should refer to each particular locking method described and to *Table L3-1* below for information about these relationships.

Exclusive Locks

An exclusive lock is the most typical lock. When you set an exclusive lock, no other user can set a lock *of the same level* (table or row) on the table or row you have locked.

If you have applied an exclusive table lock, another workstation may still be able to place an exclusive row lock on a row in the table. See "Coordinated locks" below for details.

Shared Locks

When you have a shared lock set, other users can also set a shared lock. However, you cannot apply an exclusive lock while a shared lock is set. Similarly, you cannot set a shared lock while an exclusive lock is set.

You should not use shared locks if you intend to update a table or row. Instead, use a shared lock when you need to read to a table or row (such as for a report), and it is important that no other user set an exclusive lock at the same time.

Note Some networks do not support shared locks. If this is the case, Advanced Revelation will always set exclusive locks.

Coordinated Locks

When you set an exclusive table lock, you are preventing other users from also locking that table. Similarly, when you lock a row exclusively, other users cannot also lock that row.

However, even if you have an exclusive lock set, users may be able to lock at a different level. For example, if you have an exclusive table lock set, other users might still set exclusive row locks in that table. In other words, a table and a row lock may not be "coordinated".

A coordinated lock sets a table lock, then a row lock. Coordinated locks are used when you need to mix levels of locking.

Note Coordinated locking can be controlled using an environmental setting. See "Using coordinated locking" under "Lock options" below.

Shared coordinated locks

A shared coordinated lock first attempts to set a shared lock on the table. If this lock is successful, a shared row lock is then set as well.

Note The shared coordinated lock can only be applied to rows.

This type of lock may be used to prevent updates to a row while it is being read.

Exclusive coordinated locks

An exclusive coordinated lock first attempts to set a shared table lock. If this lock is successful, an exclusive row lock is then set as well.

Note The exclusive coordinated lock can only be applied to rows.

This type of lock may be used to prevent updates to a row when coordinated locking is enabled (the *Coordinated Locking* setting at the Global Environment window).

Lock table

The table below illustrates all possible combinations of locks, and tells you what locks will be successful when faced with a locked table or row. The attempted lock is listed vertically, with the lock already set listed horizontally.

		Lock already set					
		XFL	SFL	XRL	SRL	XCRL	SCRL
Lock	XFL	N	N	Y	Y	N	N
	SFL	N	Y	Y	Y	Y	Y
Attempted	XRL	Y	Y	N	N	N	N
	SRL	Y	Y	N	Y	N	Y
	XCRL	N	Y	N	N	N	N
	SCRL	N	Y	N	Y	N	Y

Note: "F" in this table refers to "file" (table) locking.

Establishing locking

Whenever you update tables or rows in a network version of Advanced Revelation, you should be using locks. In some cases, this is taken care of automatically. In other cases, you must indicate your preference for the type and degree of locking you wish to use. If you are an R/BASIC programmer, you must set locks explicitly.

Locking in system routines

All system utilities that update rows already have locking built into them. The type of locking they do is suited to the purpose of the utility. For example, utilities that update rows are designed to execute exclusive row locks as they process. In some cases, you may have control over the type or degree of locking that the utility achieves.

Note The documentation for individual commands in Advanced Revelation will indicate the type and degree of locking used by that utility.

The Advanced Revelation system automatically locks a row when you move to a new level of the Command window (TCL level). This protects you from accidentally making conflicting changes to your open rows. You can still access the locked row(s) from another level or environment, but you cannot save a row you have already locked (see "Locking your own locks" earlier in this chapter).

Automatic Locking

Many processes that you use in Advanced Revelation will lock table or rows automatically. For example, the Advanced Revelation editor will attempt to set an exclusive row lock on the row you are accessing.

If the editor encounters a locked row, you will see a message similar to this one:

```
rowname is in use elsewhere and cannot be saved
   Press any key to continue..
```

You can view the row at that point, but may not make any changes.

User-Optional Locks

Other processes in Advanced Revelation allow you the option of setting locks yourself. For example, the row copy utility (accessed with the Copyrow window or the TCL COPYROW command) includes an option to lock rows as they are being copied. This is the (L) option.

If you choose this option, the row copy process will attempt to set exclusive row locks on the source and target rows as they are being copied. If the row cannot be locked, you will see a message like this one:

```
Waiting until the row rowname can be locked
```

Advanced Revelation's window processor offers you even finer control over locking. Using the Customize menu option in Paint, you can set a lock preference at the *Locking* prompt.

The lock type you choose at this prompt in Paint will determine how the window will lock rows. You can choose shared, exclusive, and coordinated row locks, or no locking at all.

Note For more information on setting locks using Paint, see the chapter "Creating windows using Paint" in the *User's Guide*.

Explicit Locking

If you are an R/BASIC programmer, you can take complete control over locking. Using the R/BASIC LOCK and UNLOCK commands, you can specify your preference of table or row, and shared, exclusive or coordinated locks.

In addition, you can take control over what happens when a locked table or row is encountered. For example, you can post messages and wait for user response, skip the update, wait and try again, or devise another strategy for accommodating locks.

Note For information on locking in R/BASIC, see the LOCK and UNLOCK commands in the "R/BASIC Command Reference" in *R/BASIC*.

Implicit Locking

Finally, if you are using Advanced Revelation's SQL, you can take advantage of "implicit" locking. When you update tables or rows using SQL commands, you can set a "consistency level" that indicates how much locking SQL is to do during the update.

Note Implicit locking is discussed in detail under "Implicit Locking" later in this chapter.

Setting locking options

In addition to specifying the level and degree of locking, there are some optional parameters you can use to manage locking.

Limiting available locks

Networks vary in the number of locks that may be set. For example, a network may set a limit of 400 concurrent locks for all users on the network, or may set a limit based on how much memory is available in the file server.

If you want to conserve or limit the locks available to a workstation, use the *Lock Limit* setting at the Workstation Hardware Environment window.

Note Refer to your network documentation for the maximum number of locks available.

Setting *Lock Limit* to 0 (zero) disables lock limiting. The workstation may then use the maximum number of locks available to the network. When the limit for a station is exceeded, the next lock fails. If you are programming in R/BASIC, you can detect this error using the system variable @FILE.ERROR.

Unlocking transactions at commit

For tables with transaction control applied (including those being used with SQL commands), you can choose to release all locks when a workstation commits transactions. Because of the reduced checking and coordination involved, performance improves considerably.

However, *all* locks that you have set are released, whether explicit or implicit, and whether set by the system or by you. The Advanced Revelation system locks rows in all environments whenever you temporarily exit the row (to the Command window, for example). To avoid the possibility of releasing these locks, you should not have open rows that may be affected. If you wish to guarantee maximum control over unlocking, you should *not* use this setting.

Use the *Unlock All at Commit* setting at the Concurrency Control Environment window to enable the unlock process.

Using coordinated locking

You can apply coordinated locking globally, for all workstations, by setting the *Coordinated Locking* prompt at the Global Environment window. When this prompt is set to "Y", all row locks will automatically attempt coordinated locks. If the prompt is

set to "N", table and row locks are attempted individually, with no checks made for the other level of locking.

Note See "Coordinated locks" under "Types of locking" above.

Implicit locking

In Advanced Revelation SQL, locking and concurrency controls are automatically (implicitly) applied. Although explicit locking is available throughout Advanced Revelation, implicit locking and concurrency controls are unique to SQL.

Concurrency Controls

Concurrency controls are a way to establish settings for locking and data integrity using a single parameter. The controls provided in Advanced Revelation SQL enable you to determine:

- Your choice of table or row locking
- Level of implicit locking (consistency levels)
- Deadlock detection and handling
- Lock waiting

Table and Row Locks

The prompt *SQL Implicit Locking* in the Global Environment window determines whether SQL will lock tables or rows. If the prompt reads "Row", rows are locked individually before each SQL update. If you enter "Table", SQL will lock entire tables before updating them.

Note If you choose row locking, you may run out of locks for your workstation or on your network, especially if you are using transaction processing. See "Limiting available locks" above.

Setting Consistency Levels

"Consistency level" refers to your ability to define the level of locking used and the duration of the lock (when locks are released). Five settings allow you to specify a level from no locking at all to a setting that guarantees exclusive access to data for the duration of a transaction.

Control through consistency levels is dependent on four factors. These are:

- Table and row locks. See "Table and row locking" above for details.
- A distinction between *update* operations (UPDATE, INSERT, DELETE) and a *read* operation (SELECT). An update operation makes changes to a table, while a read operation simply reads rows.
- A distinction in duration between a *query* and a *transaction*. A query is the length of time required to execute any single SQL statement. A transaction remains

active until it is committed or rolled back — this may last for the duration of many queries.

- A distinction between shared and exclusive locks. See "Degrees of locks" earlier in this chapter for information.

You can set consistency levels at the Concurrency Control Environment window. The default consistency level is assigned at logon time based on this value. You can use the SQL command SET TRANSACTION to change the default value for the remainder of the logon session. The following are the settings available for consistency level:

Level 0 A setting of zero completely disables any implicit locking done by SQL. In order to lock rows and tables, you must lock them explicitly using R/BASIC commands. You should not choose this consistency level unless you are working with a single-user system.

Level 1 A consistency level setting of 1 causes SQL to set exclusive locks during *updates* of tables or rows. The lock is held for the duration of the query. No locking takes place during reads.

Operation	Lock Degree	Duration
Update	Exclusive	Query
Read	none	none

Level 2 A setting of 2 sets exclusive locking during *updates* for the duration of each transaction. When the transaction is committed or rolled back, the locks are released. No locking is done for read operations.

Operation	Lock Degree	Duration
Update	Exclusive	Transaction
Read	none	none

Level 3 A setting of 3 sets exclusive locks during *updates* for the duration of each *transaction*. In addition, it sets shared locks during *reads* for the duration of each *query*.

Operation	Lock Degree	Duration
Update	Exclusive	Transaction
Read	Shared	Query

Depending on whether you are using table or row locking, and if you are using coordinated locking, a consistency level of 3 might provide protection against re-reading rows that have changed during your transaction.

Level 4 A setting of 4 sets exclusive table locks during *updates* for the duration of each transaction, and shared table locks during *queries* for the duration of the transaction.

Operation	Lock Degree	Duration
Update	Exclusive	Transaction
Read	Shared	Transaction

This level, the most stringent one, guarantees that other users cannot make changes to any table used or locked by the current transaction. No rows can change, and no new rows can be added to a table by another user when the consistency level is set to 4.

Because a level of 4 can potentially use a large number of locks, it is only available if you are doing table locks. The *SQL Implicit Locking* prompt at the Global Environment window must be set for table locking to use this level.

You can guarantee that transactions are processed chrono- logically when *Commit Protection* is set at the Global Environment window. This is the only level of consistency that can provide chronological commit operations.

Detecting Deadlocks

A deadlock situation occurs when two or more workstations each attempt to lock data that the other already has locked. For example, if the first user has row A locked and attempts to lock row B, while the second user has row B locked and attempts to lock A, a deadlock situation is created.

The implicit locking scheme in SQL can determine that you are involved in a deadlock situation. There are options that you can specify as well that determine how the deadlock will be resolved.

The system does not provide any automatic deadlock detection for explicit locking. However, using R/BASIC you can identify and manage deadlock situations as you wish. For more information, see "Programming with R/BASIC" in *R/BASIC*.

Note The deadlock detection process will ignore coordinated locks.

Deadlock posting

If the implicit locking system suspects that it has encountered a deadlock, it "posts" its own lock information in a central table. This table accumulates information about all workstations involved in the deadlock until the system can determine how to resolve the conflict.

You can specify the amount of time a workstation waits for a lock before posting itself to the central deadlock table. Once this information is posted, the deadlock checking process (see below) begins testing for a deadlock situation.

Use the *Deadlock Posting* prompt at the Concurrency Control Environment window to specify an optimum length of time before the posting occurs. The default value for

Deadlock Posting is 15 seconds. If you set *Deadlock Posting* to 0, the deadlock checking process is disabled.

Deadlock checking

After a workstation posts the lock to the deadlock table, the deadlock checking process is enabled. The system checks the deadlock table at specified intervals in order to determine whether a deadlock is indeed in progress.

The *Deadlock Checking* prompt at the Concurrency Control Environment window specifies the time interval used by the deadlock checking process to test whether a deadlock is in effect. The default value for *Deadlock Checking* is 10 seconds.

Deadlock resolution

When a deadlock is indeed in progress, one of the stations involved in the deadlock must give up its locks and roll back its transaction. The *Deadlock Resolution* prompt at the Global Environment window determines how a workstation determines if it must give up its locks.

Note Each workstation may select only itself as a victim.

The settings you can specify for *Deadlock Resolution* are:

- Least Active
- Youngest

Least Active. The Least Active setting uses an activity index to determine the selection. The index is based on the amount and type of I/O activity, with greater value accorded to write operations. The station with the lowest activity index becomes the victim, rolling back its transaction and releasing its locks.

Note The value for the activity index and the current transaction timestamp can be displayed with the TCL TRANSACTION STATUS command.

Youngest. The Youngest setting uses the time stamp for the transactions to identify the most recent transaction and determine the selection. If you choose this method of victim selection, the workstation that began its transaction most recently will roll back its transaction and release its locks.

The lock wait process

Often a workstation will attempt to lock data that is already locked. If a station is attempting to set an exclusive lock, and is continually defeated by the shared locks of other stations, this is called a "livelock". The only solutions in this situation are to wait until the lock is released or to quit the process and roll back the transaction.

Automatically ending lock waiting

You can specify a length of time to automatically end the lock wait process. To do so, use the *Lock Timeout* prompt at the Concurrency Control Environment window.

At the end of this time, the SQL command will be aborted. You then have the option to commit or roll back, ending the current transaction. You can also choose to ignore, which aborts the current command but does not affect the transaction.

If you are using embedded SQL, the command will return with the SQLSTATE value set to indicate a failed lock due to an interruption.

Manually ending lock waiting

Rather than continuing to wait for a lock, you can choose to manually interrupt the lock wait process (unless this is disabled, see below). By pressing [Esc] while waiting for a lock, you display the lock waiting popup. From there you have the same choices as described above.

Disabling the manual interrupt

If you do not want a workstation to interrupt the lock wait process, you can disable the [Esc] interrupt. To do so, set the *Disable Lock Interrupt* prompt at the Concurrency Control Environment window to "Yes". The default value for *Disable Lock Interrupt* is "No".

Additional lock issues

Listed below are additional topics concerning the management of locking in Advanced Revelation.

Coordinating locking with multiple file servers

Advanced Revelation does not restrict lock management to a single file server. However, if you intend to use multiple file servers, be certain that:

- Your network supports lock coordination with multiple servers.
- You attach all accessible file servers before you first log on to Advanced Revelation.
- You make sure that all users on all servers are using the same global settings and locations. All users *must* access the same Commit Log volume.

Note File ServersRefer to your network documentation for information about attaching, accessing, and locking with multiple file servers.

Locking data in bonded environments

The ability to manage locking in a bonded environment depends solely upon the bond implementation. Some bonds can coordinate table and row locking with other Advanced Revelation users, and some can even coordinate locks with users in the other environment. Refer to your bond documentation for this information.

Note See also the chapter "Introduction to Environmental Bonding".

Sharing data among copies of Advanced Revelation

If you intend to use more than one Advanced Revelation system on your network and if users will access the same data tables from different copies of Advanced Revelation (using different AREV.EXE files), note the following warnings:

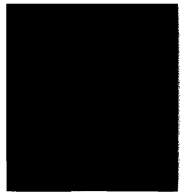
- *Do not* attempt to share data between a single-user copy of Advanced Revelation for DOS and *any* other copy of Advanced Revelation. If you use a single-user copy of Advanced Revelation for DOS to access tables, you cannot lock tables or rows. As a result, your access to data can conflict with that of other users, and the result can be data corruption.
- *Do not* attempt to share data among your Advanced Revelation systems unless the LAN Pack documentation for your network specifically states that you can do so, or you will corrupt your data. For some networks, you must complete special procedures in order to coordinate row locking among Advanced Revelation systems. See your LAN Pack network documentation for details.

Caution! Failure to comply with these stipulations can result in data corruption.

Changing network drive mappings

Do not change or set new network drive mappings for Advanced Revelation tables while you are temporarily suspended from the system. If you execute PC or SUSPEND and then change your drive mapping, the system will not lock data until you log out of Advanced Revelation and log back in.

Environmental Bonding



31. Introduction to Environmental Bonding

Environmental Bonding

Almost every computer product or system you use maintains its data in a unique, proprietary format. If you use two word processing programs, for example, you know that each stores its documents in a special format. More often than not, the formats are incompatible, and neither word processor can read documents created by the other.

This problem pervades the computer world. Each database product uses a unique format to store rows, as does each spreadsheet. If you want to access other computer systems such as minicomputers or mainframes, there is yet another layer of complexity to overcome in reading and writing data.

Sharing data

Normally, sharing data between different products or systems requires importing and exporting data. This process consists of using a special process to make a copy of the data in a new format, one that your product or system can read.

There are several problems with this solution, however. First, importing and exporting data is time consuming. Because you are making a copy of the data, it also occupies extra disk space.

More importantly, if you make a copy of a spreadsheet or database, it becomes difficult to guarantee the integrity of your data. Suppose you export a database to another product. If changes are made in both copies, which copy is current? Unfortunately, the answer is neither.

Advanced Revelation's Environmental Bonding overcomes these problems. An Environmental Bond is a module in Advanced Revelation that gives you direct access to data in its "native" format. Rather than importing the data into Advanced Revelation, you can simply bond to the environment in which the data resides. You can then read and write data directly in that format, sharing it with other users of that environment.

Your applications can use bonded tables very much as if they were Advanced Revelation tables. An application simply uses Advanced Revelation tools and utilities (windows, SQL, R/BASIC) as always. The environmental bond handles all the work of interacting with the foreign environment, giving your applications a uniform view of data.

The direct access offered by Environmental Bonding offers you these advantages:

- There is no time wasted in converting data from one format to another.
- Because you don't copy data from format to format, you save disk space.
- You can use a single set of DBMS tools — including Advanced Revelation windows, R/BASIC, and SQL — to access data in any format and environment.

- Finally, you maintain the integrity of your data. Because there is only one copy of the data, there is no danger that users might update the "wrong" copy of a database.

A new way to manage data

The advantages just listed, however, focus only on technical aspects of data access. In fact, Environmental Bonding introduces an entirely new way of thinking about data management:

- You can take advantage of the capabilities of a specific environment. For example, you can bond to a database server to get optimum speed for your queries.
- You open new horizons for your existing applications, which can now access data never before available to them.
- You can even extend the entire concept of "database". Environmental Bonding allows you to bond to such diverse environments as spreadsheets, word processors, and even CAD drawings. This gives you relational database power in areas that have not traditionally been thought of as databases at all.
- Finally, Environmental Bonding makes the PC the central focus of application development throughout your computer systems. Environmental Bonding allows you to use a single environment — Advanced Revelation — to create and maintain applications for all your computing needs, regardless of where and how the data you require is stored.

Tenets of Environmental Bonding

In the world of information management, there is a vast diversity of table types, data structures, storage methods, access techniques, and capabilities. The challenge for Environmental Bonding is to level out this diversity and overcome the differences between filing systems. Ideally, the result is a completely uniform view of data, regardless of source or location.

To meet this challenge, Environmental Bonding must address a variety of issues.

Intra-bond data integrity

The first goal of Environmental Bonding is to share data transparently with foreign environments. Given any foreign database, an Advanced Revelation application can read each row, write it back without making changes, and result in a database that is indistinguishable from the original. Tables updated through an Advanced Revelation bond can be used by applications in the foreign environment as if the changes had been made in the foreign environment.

If this level of transparent update is not possible, the bond should tell the user why a particular update operation would have resulted in data loss and therefore was not performed. Alternatively, the user can request that Advanced Revelation update the table anyway, recognizing that there will be some data loss as a result.

Inter-bond data integrity

A second goal of Environmental Bonding is to guarantee that data can be moved between any two table types without data loss. Given any two foreign environments, an Advanced Revelation application can copy a row from one environment to the other and then back to the original table. The data should be indistinguishable from the original, even by applications in the foreign environment.

If data cannot be copied without data loss (the target table might not accept a column due to its length, for example) the bond must tell the user why a particular update operation would have resulted in data loss and therefore was not performed. As with intra-bond integrity, the user can override Advanced Revelation and request that the table be updated even if data might be lost.

Intelligent data access

If a foreign environment supports high-level access methods such as SQL, the bond to that environment is written to take advantage of them. For example, bonds to SQL-based database servers can pass SQL queries to the server for processing.

No matter what high-level access methods are available, a bond always includes the ability to read and write data one row at a time. If the bond cannot use a higher-level access method, it will automatically revert to traditional row access, without intervention by the user.

Transparent table creation

Using Environmental Bonding, it is possible to write an Advanced Revelation application in such a way that it can create the tables it requires in any foreign environment. The tables can be used within the foreign environment as if they had been created in that environment.

If a table cannot be created, Environmental Bonding will tell the user which of the requirements could not be supported by the foreign environment (for example, maximum column length or lack of numeric precision).

Using an Environmental Bond

The steps described below provide very general guidelines for using any Environmental Bond. Review the steps carefully so you understand what you must do to access data in foreign tables.

Note Always carefully read the documentation that accompanies your bond. Although the steps listed below are general and apply to all bonds, the documentation for a specific bond contains valuable additional information that you will need before using that bond.

Obtaining a bond

To access data in another environment from Advanced Revelation, you must have a bond designed for that environment. Some bonds have been developed by Revelation Technologies, including bonds to dBASE, ASCII, MS-SQL Server, DB2, and Oracle tables.

Bonds to Lotus 1-2-3, Btrieve, and CAD files, among others, are available through third-party developers. Developers have created bonds to a wide variety of environments, from PCs to minicomputers and mainframes. Information about third-party bonds is available through Revelation Technologies.

If you are a developer, you can also create your own bond. Revelation Technologies will make available to developers a bonding kit containing documentation on Environmental Bonding technology, as well as sample bond programs.

Installing and configuring a bond

Once you have obtained the bond, you must install it onto your Advanced Revelation system. There may also be additional steps required to configure your system properly to use the bond.

Installing the Bond

Most bonds can be installed using the installation procedure for optional modules. For information on the installation procedure, see the chapter "Updating and enhancing your Advanced Revelation system".

Bonds available from third-party developers may use different installation procedures. Be sure to read the documentation that accompanies your bond for specific installation instructions.

Configuring the Bond

Some Environmental Bonds require additional steps beyond simply installing the bond. You may need to make changes to existing rows in your system, or you may need to know about alternate logon or other options. Be sure to review the bond documentation carefully for information about additional configuration requirements for your bond.

Displaying Installed Bonds

If your bond uses the standard Advanced Revelation optional module installation procedure, you can easily get a listing of what bonds are installed. The Who window displays a list of all the optional modules you have installed, including any bonds. To display the Who window, choose the menu option **Help-System info**.

Establishing volumes

Because Advanced Revelation can use so many different table types, it must have a way to distinguish them when accessing data. The method used is to designate a "volume" when first accessing a table.

A volume is actually two pieces of information in one: the table type and the location of the data. You can create any number of volumes to designate all the combinations of table types and locations that you require for your applications.

Note You can find a detailed description of volumes in the chapter "Managing volumes"

Before using a bond to define a new table or to access an existing table, you must initialize a volume. Follow these steps:

1. Use the option **DB Admin-Location-Define** from the Development menu. The Set Volume window displays.
2. At the *Volume Name* prompt, enter a name for the volume you are establishing. Do not use spaces in the name.
3. At the *Table Type* prompt, enter the name of the table type or bond you are using. Press [F2] or click <Options> to see a list of available table types. For example, if you will be accessing data in a dBASE file, enter DBASE_BFS.
4. At the *Data Location* prompt, enter the name of the location where your tables are or where you want to create them. In most cases, this will be a drive or path name. For remote servers, this may specify the database you want to access.
5. At the *Control Tables Location* prompt, enter the drive or subdirectory name where you want to store supporting tables such as dictionaries. Under most circumstances you can skip this prompt and allow the bond to create the tables in a default location.

   Note Control tables are described in detail under the topic "Control tables" later in this chapter.

6. Press [F9] or click on <Save> to save your volume specifications.

Whenever you use this volume name, Advanced Revelation will know both the location and table type of the tables you are accessing.

Defining new tables

Once you have established a volume for bonded tables, you can define new tables. The bond creates tables in the format of the environment it is designed to access. For example, if you define a new table using the dBASE bond, the new table will be a dBASE file.

The table definition process for a bonded table is virtually identical to that for an Advanced Revelation table. You can define new tables in a variety of ways. A common way is to use the Maketable window or the TCL MAKETABLE command at the Command window. The Maketable window is described in the chapter "Managing tables" the *User's Guide.*

You can also use Advanced Revelation's Paint utility to define a new table. Paint lets you create an entry window at the same time that you are defining a new table. Paint is described in detail in the chapter "Creating windows using Paint". If you use SQL, you can use the CREATE TABLE command to define a new table.

Although there are different methods to define and create a table, the end result is the same. Choose the method that you prefer. For example, if you are unsure of all the options you have when defining a table, use the Maketable window. If you are familiar with SQL, you might choose the SQL CREATE TABLE command instead.

Regardless of what tool you use to define a new table, the following steps describe *generally* how to define a bonded table. Be sure to read the documentation about each tool for specifics about defining a new table.

1. Locate and display the appropriate table definition tool. The Maketable window is available through the options **DB Admin-Table-Define** from the Development menu. Paint is available through the options **Define-Window**. You can enter SQL commands at the SQL window. Remember, you can use the SQL Assistant if you do not remember the syntax for CREATE TABLE.
2. When you are prompted for a volume (for SQL's CREATE TABLE command, this is the ON prompt), fill in the name of a volume associated with your bond. In most cases you can press [F2] or click <Options> to see a list of available volumes.
3. Some bonds may allow you to specify a set of "table attributes". The attributes allow you to indicate bond-specific options for your tables.
4. Fill in information about the columns you are defining in the table. This typically includes the names of the columns and their data types. Use the [F1] key to display online help, and [F2] or <Options> for the available options at any prompt.
5. Save your new definition by pressing [F9] or clicking <Save>.

Attaching bonded tables

Whether you have defined new tables or are using pre-existing ones, you must attach the tables before accessing them. (If you have just finished creating the table, it is already attached, but you will need to attach it the next time you log on to Advanced Revelation.) As with defining a table, this process is the same for bonded tables and Advanced Revelation tables.

You can attach all the tables on a volume, or individual tables. To attach a volume or list of tables, use the TCL command ATTACHTABLE, or follow these steps:

1. Display the Attachtable window using the options **DB Admin-Table-Attach** from the Development menu.
2. At the *Volume* prompt, enter the name of a volume you created earlier for the bond. Press [F2] or click <Options> to see a list of volumes that are available.
3. If you intend to attach only specific tables, you can enter the table names at the Table list prompt. Press [F2] or click <Options> to see a list of tables that are available on that volume.
4. Press [F9] or click on <Save> to attach the tables.

Once you have attached the tables, you can access them by name using any Advanced Revelation process. The tables remain attached until you log off, or until you explicitly detach them.

Accessing data in bonded tables

You can use any Advanced Revelation utility to access data in bonded tables. This includes windows, the report generators, SQL, the Editor, and R/BASIC.

One of the great advantages of Environmental Bonding is that from within Advanced Revelation, most bonded tables look and act like an Advanced Revelation table. When you are using a window, an R/LIST or SQL report, or a program, it will not be apparent to you that the data is actually stored in a bonded table.

Under some circumstances, you might encounter behavior in a bonded table that makes it different from an Advanced Revelation table. The situations in which this might occur are discussed later in this chapter under "Features and characteristics of Environmental Bonding".

Using additional bond options

Besides the basic functions of a bond — reading and writing data to tables — an Environmental Bond may offer you additional options. For example, a bond may include extended index functions, downloadable processes, or other features not available through the usual table access methods.

You can access any additional options for a bond at the Bonds menu. To display this menu, use the options **Utilities-Bonds** from the Development menu. The Bonds menu has one entry for each bond that supports additional options.

Characteristics of Environmental Bonding

One of the features of Environmental Bonding is that a bonded table behaves very much like an Advanced Revelation table. As you work with Environmental Bonding, however, you may encounter differences in features and characteristics for each table type.

The section below describes the areas in which the behavior of a bonded table might diverge from that of an Advanced Revelation table. Because each bond is different, the discussion is necessarily general. For more detailed information about the topics that appear below, be sure to review the documentation that accompanies your bond.

Sources of differences

There are two reasons why an Environmental Bond might exhibit behavior different from that of an Advanced Revelation table:

- The architecture of the foreign environment dictates that the bond must behave a certain way.
- The developer who implemented the bond has made conscious choices about the characteristics of the bond.

Differences in architecture

There may be certain Advanced Revelation features that are simply not available in the foreign environment. For example, some table types impose a row-length limit that is considerably less than the 64K available to Advanced Revelation rows. Others may not support variable-length columns, while still others may have a limit on how many columns can be defined in the dictionary.

An environmental bond will always alert you if you have encountered a difference of this sort. Normally, any operation you attempt that conflicts with the capabilities of a bond will produce an error. You may have the option of indicating in advance that you will allow certain functions, such as the truncation of data that is too long for a column or row.

Because these differences are inherent in the environment you are bonded to, Advanced Revelation cannot overcome them. You should therefore be aware of possible differences between the foreign environment and Advanced Revelation before bonding tables.

Conversely, of course, a bond may offer features that you do not normally have in Advanced Revelation. For example, some table types allow rows even greater than 64K. Others may offer extensions to indexing or other features. Depending on the bond and the feature, these may or may not be available to you within Advanced Revelation.

Differences in design

Another source of differences between bonds lies not in the table type itself, but in how it was developed. When a developer creates a bond, he or she is free to implement an elaborate bond or a simple one.

An example is a read-only bond — one that allows you to read data from a table, but not update rows. In this case, the fact that you cannot write a row is not inherent in the table type, but in how the developer chose to create the bond.

There are perfectly legitimate reasons to create bonds with these characteristics. In some cases, the developer has a special purpose in mind for the bond, and does not require that the bond be capable of all operations.

Applications

Whenever you create an Advanced Revelation table, it is associated with an application. This is usually the current application when you create the table.

Tables from other environments are not *initially* associated with a particular application. However, because applications are required for all tables in Advanced Revelation, Environmental Bonding assigns a bonded table to an application using these rules:

- If the bonded table is created within Advanced Revelation, it is assigned to the current application.
- If the bonded table already existed in the foreign environment when it was first attached, it is assigned to the GLOBAL application.

Table and column names

When you name a table or column in Advanced Revelation, there are few restrictions on creating the name. For example, you can use most any combination of letters and numbers.

Note You can find information about table and column name conventions for Advanced Revelation in the "Managing tables " chapter of the *User's Guide.*

Foreign table and column names

When you use Advanced Revelation tools to create bonded tables and columns, you are free to name them anything you like (within the constraints of Advanced Revelation naming conventions). However, this may conflict with rules that a particular bond has about names.

For example, in some cases the name of a table is limited to eight characters. This is usually the case in systems that use operating system tables directly in their applications. There are sometimes similar restrictions on column names.

To avoid a conflict like this, an environmental bond will create or suggest an alternative foreign name for the table or column. This name will fall within the naming rules for the bond you are working with. You can override this suggested name with your own foreign name. However, your name must still follow the naming rules for that bond.

When you have assigned a foreign name, the table or column will then have *two* names:

- The name that you can use while accessing the table or column within Advanced Revelation.
- The foreign name by which the table or column is known in the foreign environment.

Control tables

When you bond to a table from another environment, Advanced Revelation creates and maintains one or more "control tables". These tables are required by Advanced Revelation, but are not used by the foreign environment.

One type of control table is a dictionary. No matter what type of environment you bond to, Advanced Revelation will always create and maintain an Advanced Revelation-style dictionary for each bonded table. Another control table is the volume directory that Advanced Revelation maintains for each volume.

If you establish Advanced Revelation indexing for your bonded table, the index will also be a control table. However, if you establish an index using the capabilities of the foreign environment, the index is *not* considered a control table — instead, it belongs to that foreign environment.

For example, if you create an index for a dBASE file using the dBASE SET INDEX command, the index you create will be a part of the dBASE environment. dBASE users will be able to use and maintain the index. Conversely, if you use standard Advanced Revelation indexing tools to create an index for an ASCII file, the index is internal to Advanced Revelation and is considered to be a control table.

Note See the bond documentation for your bond to learn more about the indexing capability of the bond.

Specific bonds may also maintain additional control tables. For information about the tables and rows that a bond creates and uses, see the documentation for that bond.

Because the control tables for a bond are internal to Advanced Revelation, they are created as Advanced Revelation tables. This means that when you access data in a bonded table, you will actually be using *two* table types: the foreign table for the data, and Advanced Revelation tables for the dictionary and indexes. This is all transparent to you, however, as you use your bond. The bond itself will handle details of coordinating access to all the tables it requires.

Volumes and control table volumes

Before you can access bonded tables, you must create a volume to identify their location and table type.

Note For information on creating volumes, see the chapter "Managing volumes" in the *User's Guide*. Establishing a volume for a bonded table is also described briefly earlier in this chapter under "Using an Environmental Bond".

Environmental Bonds also typically require that you indicate a separate location for control tables. If you use Advanced Revelation tables, these control tables reside in the same location as the data table itself. Bonds, however, allow you to specify an alternative location for these tables.

The intent of specifying an alternative location is to avoid mixing the control tables with the data tables. Because the control tables are Advanced Revelation tables, you might not want to store them in the same location as tables of a different table type. For example, you might not want to store Advanced Revelation control tables in the same subdirectory as your dBASE files.

In fact, you may not even be able to store them in the same location as your data tables. If you are accessing a read-only table, for instance, the foreign environment may not allow you to create new tables in the same location.

By keeping your data and control tables separate, you do not introduce Advanced Revelation tables into a foreign environment. Separate control tables also make it easier for you to make independent backups of your data tables and control tables.

You can choose any location for your control tables. Normally, each bond will assume a default location, usually a subdirectory beneath the current directory. Wherever you keep your control tables, remember to back up that subdirectory or drive if you make a change to a dictionary, index, or other control table.

Volume directories for bonds

One of the control tables that a bond maintains is the volume directory. This tells Advanced Revelation what tables are available on that volume.

A bond builds and maintains the volume directory automatically. Each time you access the volume (by listing the directory or by accessing tables on that volume), the volume directory is updated. That way your bond will always know if a user in the foreign environment creates new tables or deletes existing ones.

Table and row structure

Advanced Revelation tables and rows are quite flexible, allowing you to store large quantities of data in variable-length columns. Bonded table types, on the other hand, may not always offer this level of flexibility.

No matter what differences there may be between Advanced Revelation and the bonded environment, the bond will usually mediate any conflicts or incompatibilities that arise. If there is a potential for loss of information, the bond will verify that you have given "permission" for this to happen.

Note It is up to each individual bond to protect you against data loss. Check your bond documentation for information on how the bond handles potential data loss.

Columns and rows

Many bonds feature a length limitation on rows that is well below Advanced Revelation's own 64K limit. Row length limitations can range anywhere from 255 bytes to 4K or more. Some bonds actually allow rows greater than 64K, although Advanced Revelation cannot take advantage of this feature.

It is also common for bonded table types to be fixed-length in some or all of their data types. Some popular table types, for instance, use fixed-length columns for most data types, but allow you to create a special "text" or "notes" column that can contain variable amounts of data.

Most bonded table types do not directly support multivalued columns, as Advanced Revelation does. On the other hand, most will permit you to store data that contains multivalue delimiters in a text-type column.

When constructing or accessing rows and columns in a bonded table, it is important that you are aware of the characteristics of that table type. For example, if you can construct a text-type column, you will want to know if you can search that column using string searches (for example, "with the word 'backorder' in the comments column"), or if you can store multivalued data in it.

As another example, in some environments a text-type column automatically occupies 4K, whether there is data in it or not. Information such as this may be important to you in designing your applications.

Tables

Advanced Revelation table types use a "hashed key" method of storage. This method is very efficient for random access of data, and allows Advanced Revelation to maintain variable-length data.

Other table types use different methods for storing and retrieving data. Of course, one of the major goals of Environmental Bonding is to accommodate the differences in the ways various environments store data. The result is that bonded tables are treated as if they were Advanced Revelation tables, with all differences resolved behind the scenes.

However, because of differences in table structures, bonds do not always perfectly emulate Advanced Revelation tables. The topics that follow address differences between bonded tables and Advanced Revelation tables of which you should be aware.

Implicit row keys

To retrieve a row from any table in Advanced Revelation, you must always have a "primary key" for the row. This is a column in the row, such as a social security or customer number, that is guaranteed to be unique for all rows in the table. An explicit key of this type is actually a part of the row. It cannot change and still identify the same row.

Many table types, however, do not use a primary key. Instead, they use an "implicit key" that describes the row's position within the table. Rather than retrieving a row using data such as a social security number, you retrieve the row using a designation such as "the hundredth row".

The difference between implicit and explicit keys presents several issues for Environmental Bonding. First, any table type that uses an explicit key can choose which column to use as the key, and the data type of the key column. With implicit keys this is not possible. An implicit key can usually only be an integer data type, and most often, you are not allowed to specify which column is to be the key.

A more important difference is that while an explicit key never changes, an implicit key *can* change. This might happen, for instance, if a row shifts position in the table because the table was "packed" by a user in the foreign environment.

Because of this, processes that depend on fixed keys can be unreliable. A good example is an index. Because an index stores a row's key, it is out of date as soon as the key changes. For this reason, you may want to refrain from using Advanced Revelation indexes with table types that depend on implicit keys.

Note Refer to the documentation for your bond to see if the bond uses implicit keys, and how they are handled within the Advanced Revelation environment.

Dictionaries

No matter what type of table you access, there is always an Advanced Revelation dictionary for the table. In the case of bonded tables, this dictionary is created automatically by the bond.

To the extent that it is possible, Advanced Revelation builds dictionaries according to information available in the foreign environment. For example, the dBASE bond reads the structure of a dBASE file and constructs a corresponding Advanced Revelation dictionary.

For some bonds, this is not possible. An ASCII file, for instance, does not store information about its structure. As a result, Advanced Revelation cannot directly construct a dictionary for the table, because there is no way for it to deduce the table structure automatically. In that case, the user is responsible for creating a dictionary structure that matches the table.

Synchronizing dictionaries

It is always possible that the structure of a foreign table can change from outside of Advanced Revelation. A dBASE user, for instance, might change the structure of a dBASE file. If so, the Advanced Revelation dictionary for that table is no longer accurate.

To account for this possibility, a bond usually checks the correctness of the Advanced Revelation dictionary for a bonded table against the actual structure of the table. This check is done when the table is first accessed.

Synchronization error

If the structure of the foreign table has been changed outside of Advanced Revelation, you will get an error when you attempt to access the table. A message will display indicating that the structure of the table has changed since the last time you accessed the table inside Advanced Revelation.

To correct this problem, you must delete the Advanced Revelation dictionary for the bonded table. This will clear the Advanced Revelation view of the bonded table. When you next attempt to access the table, a new dictionary will be built automatically by the bond.

If you want to preserve certain columns out of the dictionary, you should copy them to a temporary table before deleting the dictionary. For example, if you have created some symbolic columns to use with your bonded table, you should copy these to a dictionary table elsewhere in your system. After the new dictionary is built by the bond, you can copy the saved columns to the new dictionary.

Deleting a dictionary

Because a bond checks the Advanced Revelation dictionary against the foreign table structure when it accesses the foreign table, you will not be able to delete an Advanced Revelation dictionary as you can for Advanced Revelation tables. If you delete the dictionary using the syntax DELETETABLE DICT *tablename*, the dictionary will indeed be deleted. However, as soon as you access the bonded table again, the dictionary will be rebuilt as part of the synchronizing process.

Column position gaps

One of the attributes that you specify for any type "F" column is the position that the column holds in the row. Normally, you would assign positions sequentially — the first column is position 1, the second column is position 2, etc.

If you are defining an Advanced Revelation table, you can define columns in any order you like. You can also assign positions arbitrarily. For example, you might have columns 1 through 3, skip position 4, and then continue with columns 5 through 11.

Bonded tables, however, rarely allow this flexibility. Especially if you are defining a fixed-length table, columns must be ordered sequentially with no gaps.

Environmental Bonding will enforce this restriction if it applies to your bond. If you are using Paint or the Dictionary window, it is possible to create columns with non-sequential positions. But when you try to create the table, the bond will refuse to do so. You must then edit the dictionary definitions, and make sure that they are all sequential.

Restructuring tables

Because of the variable-length nature of Advanced Revelation tables, there is normally no restriction on making changes to the structure of a table. For example, you can add a column at any time by simply creating a new entry in the dictionary.

Many other table types do not allow this, however. Typically, once a table is created, you cannot add new columns or remove existing ones.

To change the structure of these types of tables, you must completely rebuild the table. As a rule, Advanced Revelation leaves this to the foreign environment. For example, if you want to rebuild a bonded dBASE table, you must do so in the dBASE environment. When you reattach the dBASE file in Advanced Revelation, you can delete and recreate the Advanced Revelation dictionary for that table.

Environmental Bonding will always check a bond to see if you are allowed to make changes to the table structure. If you are not, the bond will prevent you from making the changes.

Caution! If you use Paint to change or add column definitions with a bond that does not allow you to modify the table structure, Paint will prompt you to delete the old table and create a new one. *This does not rebuild the table.* When you rebuild the table, all the data remains intact. If you use Paint to change the structure of a table, *all the data is deleted, and a new (empty) table is created.*

Deleting tables

Because an environmental bond allows you to share tables with users outside of Advanced Revelation, there are some guidelines that apply when you want to delete a table.

One of the attributes that is stored for every table is a "delete flag". You can see and change this flag in the Maketable window at the *Allow Table Delete (Y/N)?* prompt. When the delete flag is "Y," you *can* delete the data portion of the table.

The setting of this flag is determined when you create a table, or when you first attach it. Advanced Revelation uses these rules when setting the flag:

- If you create the table within Advanced Revelation, the flag will allow you to delete the table (the flag is set to "Y").
- If you attach a table that already exists in the foreign environment, the flag prevents you from deleting the table (the flag is "N").

You can change this flag if you like. For example, you might want to delete a dBASE table that already existed when you first bonded to it. To do so, use the Maketable window to set the flag to "Y", and then delete the table using the Deletetable window or the DELETETABLE command at the Command window.

Data types

Advanced Revelation provides a basic or standard set of data types for columns. These include INTEGER, CHAR, DATE, TIME, and the other data types most often available in database systems.

Each bonded environment, of course, maintains its own set of data types. Some environments may offer additional data types or variants on Advanced Revelation's standard types. Other environments may offer a more restricted set.

In order to be compatible with as many environments as possible, Environmental Bonding matches or "maps" the standard data types of Advanced Revelation against those available in the foreign environment. There are three ways that Advanced Revelation and foreign data types are matched:

- A foreign data type may map exactly to an Advanced Revelation data type.

- If the foreign environment supports fewer data types than does Advanced Revelation, more than one Advanced Revelation data type can map to a single foreign type. For example, in the dBASE bond, the Advanced Revelation types DOLLARS, INTEGER, and DECIMAL are all mapped to the dBASE N (Numeric) data type.
- The foreign environment may have more data types available than Advanced Revelation. In that case, the user has a choice of foreign data types onto which to map an Advanced Revelation data type.

Type mapping

One of the characteristic features of a bond is how it maps its data types to Advanced Revelation data types. For many applications, it is important to know the nature of data types in both environments, in order to design and implement applications correctly.

When you create a new column in a bonded table, you are actually mapping an Advanced Revelation data type to a foreign data type. For example, if you select the Advanced Revelation data type INTEGER for a column, it will map to the dBASE data type N (numeric).

You do not actually have to make the correlation between the Advanced Revelation and bonded data types. This is done automatically by the bond. You only need to choose the data type appropriate to Advanced Revelation.

To see what foreign data type has been assigned to a column, you can look in the Dictionary window. Follow these steps:

1. From the Development menu, choose the options **DB Admin-Columns-Define** to display the dictionary window.
2. Enter the names of the table and column.
3. Press the [Shift-F5] Softkey to display additional column options.
4. The entry at the Foreign data type prompt will tell you what foreign type has been assigned to the column. This prompt is for information only; you cannot change the entry.

Data type definitions

A one-to-one mapping of data types between environments is not a guarantee of compatibility. For example, the existence of a type INTEGER in both Advanced Revelation and in the foreign environment does not mean that an *integer value will* transfer successfully between environments.

It is important to know the *precise* definition of a data type in each environment. For example, in some environments, an integer is defined using 32 bits. In other

environments, an integer uses 64 bits. Transferring an integer from the more precise to the less precise environment can result in some degree of data loss (see "Loss of length and precision," below). As another example, the exact definition of VARCHAR in each environment can be quite different.

The documentation for each bond gives you specific information on how each data type is mapped into the Advanced Revelation standard data types. It also lists any bond-specific data types and how these are implemented both in the foreign environment and in Advanced Revelation.

Advanced Revelation data types

In Advanced Revelation, all data types are eventually reduced to one of the three following types of data:

- An integer with 64-bit precision (32-bit precision in Advanced Revelation for OS/2).
- A floating-point number with precision to 15 decimal digits.
- A character string up to 64K.

The following table lists the Advanced Revelation standard data types:

Advanced Revelation Standard Data Types

Type	Data	Type	Data
BOOLEAN	integer	FLOAT	float
CHAR	char	INTEGER	integer
DATE	integer	TEXT	char
DATETIME	float	TIME	integer
DECIMAL	integer	VARBINARY	char
DOLLARS	integer	VARCHAR	char

The definitions of each of the types of data in Advanced Revelation dictate the highest possible precision and size available for a bond. If data in a foreign environment exceeds these definitions (for example, a string longer than 64K), one of two options is available:

- If data precision is important, the read or write operation that encounters the data will fail.
- You can establish permissible data loss, such as a truncated column, in order to accommodate the bonded environment. Details on potential data loss are discussed below.

Bond-specific data types

When you choose a data type for a column in a bonded table, you always have the choice of Advanced Revelation data types. For example, no matter what environment you are bonded to, you can always select the data type DATE or INTEGER. The bond will make sure that the data type you select is mapped appropriately to the foreign environment.

Bonds frequently add their own data types to the list of types available for a column. You can always tell when a data type is bond-specific, because the name of the bond precedes the type name. For example, the data type D (date) from the dBASE environment appears as the data type DBASE_D within Advanced Revelation.

Bond-specific data types give you the opportunity to choose a foreign data type directly. This is useful if you are already familiar with the data types available in that environment. In that case, you do not need to examine how an Advanced Revelation data type maps to the foreign data type. For example, if you are a dBASE user, you do not need to choose between the Advanced Revelation types INTEGER and FLOAT — instead, you can directly choose the dBASE type N (numeric).

The documentation for a bond always describes any bond-specific data types. This documentation provides you with information on the names, characteristics, and mappings of these types.

Bond-specific data types are automatically added to your Advanced Revelation system when you install a bond. You can therefore see a list of these data types whenever you are asked for data type. For example, if you are defining a dBASE file and are asked for a data type, the [F2] Options popup will list not only the Advanced Revelation standard data types, but the dBASE-specific data types as well.

Note Remember that the Advanced Revelation data types are generic and are implemented in each bond. If portability of your application is an important issue, you will want to use Advanced Revelation data types when defining a column.

Column definitions

In Advanced Revelation, the dictionary is the central repository for all information about a column. Everything from the column name and size to indexing information to user-specified attributes of a column are stored in the dictionary.

Several of these attributes are specific to bonding. If you use only Advanced Revelation tables, you will not normally use these attributes. However, when you create columns in a bonded table, you will need to provide — or at least know about — this additional information.

Master column

In Advanced Revelation tables, you can create any number of columns that describe the same data. As an example, you might create two columns that define column 1 to be a number. The first column might print the data as money, with a decimal point and a dollar sign. A second column might treat the number as a simple decimal number.

When you create columns in a bonded table, you can create these synonym columns also. However, one of the columns that describes the data must be designated as the "master column". The master column contains not only the usual information about the column (name, display length, justification), but includes all the additional information needed by the bond to access the data (data type mapping, truncation flags, etc.). By allowing only one master column, a bond prevents you from creating two or more columns that have contradictory information about how to access the bonded data.

Creating a master column

The first definition that you create for a column will become the master definition. This is true whether you use the Maketable window, the Dictionary window, Paint, or SQL to define your columns. If you define additional columns that describe the same data, they will not be considered master columns — they will be synonyms for the master.

Column attributes

Each bond can maintain any information it requires about a column. Some foreign environments define relatively simple columns, with few attributes. Others, like Advanced Revelation itself, maintain an extensive list of attributes.

For Advanced Revelation tables, the information normally stored in a dictionary definition is adequate to completely describe a column. However, in bonded tables, you must almost always include additional information.

To account for the instances where additional information is required, each bond can keep a bond-specific set of attributes in the dictionary. This is simply a list of one or more parameters required to completely define the column.

For example, in a fixed-length column, one of the column attributes might be the physical length of the column. The ASCII bond, for example, uses column attributes to store the input and output formats of the data.

You can specify column attributes in the Dictionary window. To access them, follow these steps:

1. From the Development menu, choose the options **DB Admin-Columns-Define** to display the dictionary window.

2. Enter the names of the table and column.
3. Press the [Shift-F5] Softkey to display additional column options.
4. At the Foreign column attributes prompt, press [F2] to display a column attribute window. You can fill in the column attributes according to the prompts you see. Press [F1] for help, and [F2] for options at any prompt.
5. Save your definition by pressing [F9] (Save).

Loss of length and precision

Because each environment has a different definition for its own data types, transferring data from one environment to another can result in loss of information. For example, one environment may define an integer in 64 bits, another in 32 bits. Copying an integer from the first environment into the second results in loss of precision and can therefore result in the loss of some data.

The same applies to character columns. If one environment allows 4K strings, but another is limited to 255 characters, copying data from one to the other can lose data.

Caution! Because of the limitations of foreign environments, not all Environmental Bonds provide the full protection against data loss that is described below. For more information, see the documentation for your bond.

Default protection against data loss

Environmental Bonding is designed to fail when you attempt operations that cause any loss of data. For instance, if you attempt to write a column that contains control characters into an environment that does not allow these, the write operation will fail with the error that you have violated a data type restriction.

Degrees of data loss

Although Environmental Bonding will not by default allow data loss, you can override this. When you define columns for your application, you have the option of specifying the degree of data loss that you will accept.

Note Specifying a degree of acceptable data loss does not automatically mean that data will be lost. It simply tells the bond what to do if *a particular row* contains data that is too precise or too long for the target environment.

Environmental Bonding allows you three levels of acceptable data loss:

- **None.** You will not accept any degree of data loss. If an operation is attempted that would result in any data loss, the operation is rejected. This is the default for a new column.

- **Some.** You will accept some degree of data loss, even though the preciseness or the length of the column in one environment does not match that in another. For instance, one environment may use 18 decimal digits for a floating point number, another uses only nine. You might decide that it is acceptable for digits to the right of the decimal point to be rounded, resulting in a partial loss of information.
- **Total.** You will tolerate the complete loss of information. If you attempt to store extremely large numbers in a column that is not large enough, the resulting truncation may result in total information loss.

When you specify a degree of acceptable data loss, you do so for two conditions: input and output.
Input means data that is being read into Advanced Revelation. If you encounter a potential data loss on input, it implies that Advanced Revelation supports less precision or a smaller string length than the foreign environment.

Output means data that is being written to the foreign environment. If there is a question of data loss during output, the foreign environment is more limited than Advanced Revelation.

In addition to simply setting a degree of acceptable data loss, you can also specify values that are used for "overflow" (number too large) and "underflow" (number too small) conditions. For example, you might specify that if the bond encounters a floating-point number that is too large, that it substitute the overflow value 9999.999999.

Specifying acceptable data loss

To set the degree of acceptable data loss for a particular column, follow these steps:

1. Use the option **DB Admin-Columns-Define** from the Development menu to display the Dictionary window.
2. Enter the name of the table and column that you are defining.
3. Press the [Shift-F5] Softkey to display additional column options.
4. At the Acceptable Data Loss prompt, enter a separate degree of acceptable data loss for both input and output. Use these values:

   0 No data loss
   1 Some data loss
   2 Total data loss

5. If you have entered a value of 1 or 2 for acceptable data loss, you can enter additional information. Next to the degree values for input and output, enter overflow and underflow values for the column. These values are optional.
6. Save with [F9] or by clicking <Save>.

Null handling

Some environments make a distinction between a column with no value and a column that is null. An integer column is a good example. If the column contains all zeroes, the value of the column is zero. However, if there is nothing whatsoever in the column, the column is null.

Although less common, this type of distinction is also sometimes found in character columns. In that case, there is a distinction between an "empty string" and a "null string". In the former case, some designator in the column (perhaps a string of blanks) indicates that a string of no particular value has been assigned to the column. In the latter case, no value at all is assigned to the string.

Note Advanced Revelation itself does not distinguish between an empty string and a null string. If Advanced Revelation encounters a string of length zero, it will treat it as a null string.

The fact that some environments distinguish nulls while others do not is a potential source for information loss. If you write a null column to an environment that has no concept of null, the "nullness" of the column is lost. Because Advanced Revelation does not distinguish null data from empty strings, the same problem occurs when you attempt to read null data into Advanced Revelation.

Caution! Because of the limitations of foreign environments, not all Environmental Bonds provide the full protection against data loss that is described below. For more information, see the documentation for your bond.

Accepting null data

By default, Environmental Bonding will not allow loss of information. It will therefore cause an operation to fail if the "nullness" of a column is lost in the process.

Two attributes in the dictionary definition for a column, however, permit you to specify how a bond is to handle nulls.

First, you can set a flag giving the bond permission to accept null data even if it has no way to indicate a null. In that case, the "nullness" of the column is lost. However, the bond will write an appropriate value back to the bonded table. For instance, the bond might write a string of blanks back to a character column to indicate a null.

Second, the bond may allow you to override the bond's definition of "appropriate character" as a default null value. For example, you might want to have the bond write out the literal string "NULL" to a character column in order to indicate a null value. When the bond reads the row later, it will convert the string "NULL" into a true null value.

Note Every bond establishes a default value for null for each data type. The documentation for a bond includes information about the default value for null.

To establish null handling for a column in your bond, follow these steps to allow null columns:

1. Use the option **DB Admin-Columns-Define** from the Development menu to display the Dictionary window.
2. Enter the name of the table and column that you are defining.
3. Press the [Shift-F5] Softkey to display additional column options.
4. At the Accept Null prompt, enter "Y". If this prompt is blank, or if it contains "N", the bond will not accept null data for this column.
5. At the Null Char prompt enter a value that you will recognize later as a null.

Indexing

The benefits of indexing extend to bonded tables as well. By establishing indexes for bonded tables, you can gain performance improvements in searches and sorts.

Applying Advanced Revelation indexing

In general, you should not use an Advanced Revelation index with a bonded table, because Advanced Revelation indexes are only updated by Advanced Revelation processes. If you use a process in the foreign environment to update data in a bonded table, the Advanced Revelation index on that table is not updated.

The rule of thumb you should follow is this:

- If the bonded table will be updated from outside of Advanced Revelation, you should not use an Advanced Revelation index with the table.
- If the table will be updated *only* from within Advanced Revelation, then an Advanced Revelation index is acceptable.

You should not use Advanced Revelation indexes if your bonded table type uses implicit keys. In that case, any operation that causes the implicit keys to change, such as "packing" the data, invalidates the indexes. To keep the index current, you must rebuild it after every operation that changes row keys.

Foreign indexes

Some foreign environments have their own indexing systems. For example, dBASE has a facility for creating, maintaining, and using indexes with dBASE files. If a bonded environment has its own indexing system, Advanced Revelation can take advantage of it when querying tables. In fact, you can use Advanced Revelation and foreign indexes together in accessing your bonded tables (subject to the recommendations above).

Using indexes with bonded tables

Advanced Revelation will use both Advanced Revelation and foreign indexes when querying the table. Certain functions, however, such as direct index lookups in windows, will not use a foreign index. These are the rules for how Advanced Revelation will use indexes with bonded tables:

- R/LIST and SQL will use any index (Advanced Revelation or foreign) for searching. R/LIST will also use any index for sorting. SQL will not use foreign indexes for sorting (ORDER BY), but will take advantage of them to optimize joins.
- In windows, any search method that generates an R/LIST sentence will use indexes as described above. This includes Query mode, EasyWriter, and [Ctrl-F10] filters.
- System functions that read indexes directly will use Advanced Revelation indexes, but not foreign indexes. This includes the system subroutines BTREE.EXTRACT and COLLECT.IXVALS, as well as the code and command codes "B" and "V".

Networks and locking

If you are using an Environmental Bond while on a network, you need to know what kind of table and row locking the bond supports:

- No locking.
- Locking coordinated with other Advanced Revelation users.
- Locking coordinated with users outside of Advanced Revelation.

No locking

If a bond does not provide any locking at all, you must use some other method to ensure that only one person updates a table and row at a time. One method is to set the network's "share" flag for a table so that only one user at a time can access the table. This trades the convenience of multi-user access for the security of system-wide locks.

Locking within Advanced Revelation

A bond may set locks that are respected by all Advanced Revelation users. This is important if you expect more than one Advanced Revelation user at a time to access bonded tables. However, a user from another environment can still violate a lock that was set within Advanced Revelation. To prevent this, you must use a locking method like that described above.

Locking in the foreign environment

A bond may also set locks that are respected both by Advanced Revelation users and by users in the foreign environment. The dBASE bond is a good example. If you lock a row in a dBASE file using Advanced Revelation, the lock is respected by anyone accessing the file using dBASE.

32. The ASCII Environmental Bond

Description of the ASCII bond

The ASCII bond enables you to write or read an ASCII text file directly, without going through a time-consuming data import or export process. If you run applications that generate ASCII data files, you can use the ASCII bond to access these files within Advanced Revelation. You can also use the ASCII bond to create ASCII files that both Advanced Revelation and your foreign application can access and update.

The ASCII bond works with two types of ASCII text files: those that use delimiters to indicate variable column length, and those that have fixed column lengths. Once you indicate which file format you want to use and add column definitions to the data dictionary, the bond can access and maintain data in the file based on your specifications.

Variable vs. fixed-length ASCII file

Below are examples that show the same data formatted for a variable length file and a fixed length file.

Variable-Length ASCII File Example:

```
Smart-COMM,Delores Thompson,(408) 312-3533
Computer Micro-Systems,Scott Gerson,(704) 213-8732
Allied Software,John Draftson,(415) 555-1212
Alex Service Inc,Alex Southward,(801) 853-0987
ACM Accounting,Allen Jacobson,(612) 497-2364
```

In a variable-length file, the data written to a column can be any length. The end of each column is indicated by a column delimiter. In the example above, the delimiter is a comma. Columns are located by their position in the row and each column is separated by a comma.

Fixed-length ASCII file Example:

```
Smart-COMM           Delores Thompson   (408) 312-3533
Computer Micro-Syste Scott Gerson       (704) 213-8732
Allied Software      John Draftson      (415) 555-1212
Alex Service Inc     Alex Southward     (801) 853-0987
ACM Accounting       Allen Jacobson     (612) 497-2364
```

In a fixed-length file, each column in the row has a predetermined length, which is the amount of space allocated for that column. If you write less than the maximum number of characters to a fixed-length column, the data is padded with spaces so that the next column in the row begins in its proper position. You cannot write more that the number of characters allocated for that column. For example, in the second row shown above, only part of the company name — "Computer Micro-Syste" — could fit in the first column. If you look at the second row in the variable-length file example, you see that the length of the data stored in the column is dynamic rather than fixed, so the entire company name can fit into the column.

Installation

You can install the ASCII bond using the installation procedure for optional modules. See the chapter "Installating Advanced Revelation" in the *User's Guide*.

Characteristics of the ASCII bond

The structure of ASCII files requires that you keep a few issues in mind when using the ASCII bond. Details about the characteristics listed below appear in the "Using the ASCII Bond" section.

ASCII File Types The ASCII bond enables you to create and access fixed-length and variable-length ASCII text files.

Accessing Binary Data You cannot use the ASCII bond to access ASCII files that contain data in binary format.

Illegal Delimiter Characters The ASCII bond reserves several characters for use exclusively by the bond. You should not use any of the following characters for row or column delimiters or for the text marker:

ASCII CHAR 19	
ASCII CHAR 20	
ASCII CHAR 21	
ASCII CHAR 252	(subvalue mark)
ASCII CHAR 253	(value mark)
ASCII CHAR 254	(field mark)

Implicit Keys The ASCII bond uses"implicit" keys to identify rows. The keys are implicit because they do not exist as data in the ASCII file. Instead, the row keys indicate the position of each row in the file, beginning with row key 1 for the first row and continuing sequentially.

Indexing You should apply Advanced Revelation indexing (SI.MFS) *only* to ASCII files that will not be accessed or updated by processes outside of the Advanced Revelation environment.

Maximum Row Length The ASCII bond can work with an ASCII file of any size. However, individual rows in the file may not be longer than 64K. Row length could be limited by available memory.

Locking The ASCII bond does not support row or table locking. ASCII files can be corrupted if two or more users attempt to write to the same file at the same time. Advanced Revelation users should not access ASCII files that are being used by other users in a multiuser or network environment.

Null Handling The ASCII bond makes no distinction in string columns between a null value and an empty string.

Using the ASCII bond

Working with a bonded ASCII file is very similar to using any other Advanced Revelation table. You can use most of the same tools and perform most standard operations on ASCII files.

Note The "Introduction to Environmental Bonding" chapter provides background information about standard Environmental Bond behavior.

Any requirements or distinctive operations involved in using the ASCII bond are explained below. In all other circumstances, you should assume a bonded ASCII file requires no special treatment.

Accessing ASCII files

You must create a volume for your ASCII files before using them. A volume will tell Advanced Revelation that you are using the ASCII "file type", plus the location of your ASCII files. You can create multiple volumes if you have ASCII files in more than one location.

To assign a volume, choose the **DB Admin-Location-Define** from the Development menu.or use the TCL SETLOCATION command.

Fill in the prompts as for any volume (for information on volumes, see the chapters "Managing volumes" and "Introduction to Environmental Bonding"). At the *Table type* prompt specify "ASCII_BFS" to indicate that the files in this volume are ASCII files. If you do not fill in a value at the *Control Tables Location* prompt, the ASCII bond will place the control tables in a default location (the sub- directory ASCII.BND beneath the current subdirectory).

The files in an ASCII volume must be attached in order to make them available to Advanced Revelation processes. Choose the **DB Admin-Table-Attach** options from the Development menu, or use the TCL command ATTACHTABLE.

Using existing ASCII files

ASCII files typically do not store any information about their file structure. Therefore, when you first access an existing ASCII file, Advanced Revelation creates only a rudimentary data dictionary for the file. To successfully use the ASCII bond, you *must* expand on the initial data dictionary information — providing column definitions and specifying appropriate file attributes. *This is not an automatic process.* If you do not supply this dictionary information, the ASCII bond cannot read or write to an ASCII file.

The dictionary that Advanced Revelation creates for an ASCII file is assigned the standard Advanced Revelation identifier of:

```
DICT.filename
```

where *filename* is the name of the ASCII file. To display the dictionary, use the TCL command LISTDICT.

The initial data dictionary for an ASCII file defines only one column — the implicit key column. You must make these changes to the file's data dictionary so that it accurately describes the file and row structure:

- Specify appropriate file attributes.
- Add column definitions to describe the row structure.

Caution! Once you complete the file definition process, you should test the accuracy of your current file definition by querying the file. *Do not attempt to add or delete data in an existing ASCII file unless you are confident that the current dictionary is accurate.* For more information, see "Testing the file definition" later in this chapter.

The Maketable window enables you to define the file attributes and column characteristics of an existing ASCII file. To access this window, choose the **DB Admin-Table-Define** options from the Development menu.

Note In addition to the information provided below, you can find documentation about the use of the Maketable window in the "Managing tables" chapter.

Specifying ASCII file attributes

When you access an existing ASCII file, the ASCII bond assumes that the file structure complies with a set of default file attributes:

- The file is variable-length
- The column delimiter is a comma
- The row delimiter is carriage return/linefeed

If the default attributes are not correct for a given file, various problems may arise. For example, if the file contains no carriage return/linefeed sequences, the file will appear to be one big row. In that case, if the file is bigger than the maximum row size, attempts to access the file will result in an error.

In order to read and write the rows in an ASCII file properly, appropriate file attributes *must* be specified. You must inform the ASCII bond if your ASCII file is a fixed-length file or if your delimiters differ from the standard variable-length file delimiters.

Note If a foreign application generated your ASCII files or will be accessing them, see the documentation for the foreign software to find out which file format and delimiters to use.

Changing file attributes

Make changes to the current file attribute specifications at the Maketable window. To do so:

1. At the *Tablename* prompt, enter the name of the ASCII file that you want to work with.
2. At the *Location or volume* prompt, enter the volume assigned for the file. The current dictionary information for the file will display.
3. Press [Shift-F1] to display the ASCII File Attribute window. You can press [F1] for online help at any of the prompts in the window.
4. Enter the correct file attributes at the prompts in the window and press [F9] to save your specifications. For more information about each of the prompts in the ASCII File Attribute window, see "Defining files" later in this chapter.

Defining ASCII columns

To create a new ASCII file or access an existing one, you must define all of the data columns in the row. The ASCII bond cannot recognize the individual columns until you add the appropriate column definitions to the data dictionary.

Note If a foreign application generated your ASCII files or will be accessing them, see the documentation for the foreign software to find out about the row structure and column characteristics of the file.

If you intend to make updates to a bonded ASCII file, the Advanced Revelation dictionary must accurately describe the existing file structure.

Use the lower half of the Maketable window to enter column definitions. For more information, see the "Managing tables" chapter in the *User's Guide*.

Data types

The ASCII bond supports the bond-specific ASCII_STRING data type and all of the standard Advanced Revelation data types. If the type you specify requires additional parameters, a collector window displays. Fill in the prompts, and press [F9] to save your specification.

ASCII_STRING

There is one data type, ASCII_STRING, that is unique to the ASCII bond. Unlike the standard data types, the ASCII_STRING data type has no default characteristics.

When you use the ASCII_STRING data type, a collector window displays. You must specify two parameters: justification and length. Optionally, you can specify conversions for data input and output.

Note Your ASCII_STRING parameter specifications for a column are also used as that column's default foreign column attributes.

- *Input Conversion* specifies the internal format that Advanced Revelation should convert data to when it is read from the ASCII file. See ICONV in the "R/BASIC Command Reference" chapter in the *R/BASIC* manual for a list of valid conversion specifications.
- *Output Conversion* specifies the format that Advanced Revelation should convert the data to when it is written to the ASCII file. See OCONV in the "R/BASIC Command Reference" chapter in the *R/BASIC* book for a list of valid conversion specifications.
- At the *Justification* prompt you can specify one of three ways that the data will be justified — L (left), R (right), or C (centered).
- If you are defining a fixed-length column, the *Length* parameter determines the maximum number of characters that can be stored in the column and the default display length for the column. If you are defining a variable-length column, *Length* does not limit the number of characters that can be entered in the column. Instead, it serves as a default data display specification, determining how many characters display on reports.

Specifying foreign column attributes

You can specify foreign column attributes for each column that reads and writes data to the file. The foreign column attributes define precisely how the data should be read for use in Advanced Revelation, and how it should be written to the ASCII file.

Note All data types include a set of default foreign column attributes. You only need to specify foreign column attributes if you do not want to use the defaults.

To view or change the current foreign attributes for a column, press [Shift-F3] at the Maketable window. Then enter the name of the column at the prompt. A collector window displays so that you can view and enter attribute information.

The foreign column attributes entry window and the ASCII_STRING data type parameter entry window both prompt for similar information. See "ASCII_STRING" above for details about what to enter at the *Input*, *Output*, and *Justification* prompts.

- If you are defining a fixed-length column, *Length* specifies the maximum number of characters that can be stored in the column. If you write less than the maximum number of characters to the column, the data is padded with spaces to equal your specification. For variable-length columns, you can use the *Length* specification to determine the minimum number of characters that can be stored in the column. If you write less than the minimum number of characters to the columns, the data is padded with spaces to equal your specification. If you write more than the minimum number of characters to the column, the *Length* specification is ignored.

- If you are defining a variable-length column, you can enter Y at *Use text marker (Y/N)?* to specify that data should be surrounded by the text marker when it is written to the file. The character used as a text marker is specified as part of the file attributes. A " (quote) character is the default text marker.

Note For more information about text markers, see "Specifying ASCII file attributes" later in this chapter.

Testing the file definition

To determine whether or not your current file definition is accurate, use R/LIST to view the contents of several rows in the file. To do so:

1. Press [F5] to display the Command window.
2. Enter a LIST command, specifying how many rows to list and naming all of the columns in the row. The command syntax is:

   ```
   LIST {n} filename columnname(s)
   ```

   For example, this command lists 5 rows in a CONTACTS file that has columns for name, city, state, and phone number:

   ```
   LIST 5 CONTACTS NAME CITY STATE PHONE
   ```

The report will display in the View window. Each column following the key represents a column in the file. Just by looking at the data displayed in the report columns, you should be able to determine whether your current definition of the file structure is accurate.

If data appears in the wrong columns:

- Fixed-length file — Did you specify the column lengths incorrectly? Is your row delimiter correct? Did you define all of the columns in the row?
- Variable-length file — Are you using the proper file attributes for row and column delimiters? Are you using a text marker correctly? Did you define all of the columns in the row?

Defining files

To create a new ASCII file, choose the **DB Admin-Table-Define** options from the Development menu, or use the TCL MAKETABLE command. Refer to the "TCL command reference" chapter in the *Reference* manual. In addition to the information provided here, you can find documentation about the use of the Maketable window in the "Managing tables" chapter in the *User's Guide*.

File names

Although the Advanced Revelation name for your bonded file can be any valid Advanced Revelation identifier, this name may not be a valid file name at the operating system level.

If you are creating an ASCII file and your filename specification is illegal, the ASCII bond will display a message. The message indicates that the bond has assigned an alternate *foreign* filename for use outside Advanced Revelation:

Press [Enter] to accept the assigned foreign name or enter the foreign name you prefer. The file name that you enter must use the standard DOS file naming conventions:

- a filename of one to eight characters
- a dot (".")
- a filename extension of one to three characters

If you do not specify a filename extension when creating a new file, the bond automatically assigns the extension ".DAT" to the foreign filename.

Specifying ASCII file attributes

Use the *Table Attributes* prompt in the Maketable window to specify the correct file attributes.

Press [F2] at the *Table Attributes* prompt to display the ASCII File Attribute window. Attributes are entered at the following prompts:

File type

You must specify whether the ASCII file is fixed-length or variable-length. Enter F for fixed, V for variable. The default is V.

> **Note** If you specify an F (fixed-length) file type, the prompts for *Column Delim*, *Value Delim*, and *Text Marker* are removed from the window.

Header cnt

To define a file header at the beginning of the ASCII file, enter the number of bytes to reserve for the header. A specification at this prompt tells the ASCII bond not to write to this portion of the file. Only data that is stored after the header is accessible by the ASCII bond.

Row delim

The row delimiter is required when you are defining an ASCII file that uses variable column length row format. It specifies the ASCII character used in the file to indicate the end of a row. The row delimiter may be 1 or 2 bytes in length. The default delimiter is a carriage return and linefeed (13:10).

Enter the ASCII decimal number of the row delimiter or press [F2] or click <Options> to display the ASCII Character Set popup, which lists all ASCII characters. If the row delimiter is 2 bytes, separate the two ASCII characters with a : (colon). For example, for a row delimiter of carriage return-linefeed, enter:

```
13:10
```

Column delim

The column delimiter is required when you are defining an ASCII file that uses variable-length columns. The column delimiter specifies the ASCII character that indicates the end of a column. The column delimiter is limited to 1 byte in length. The default delimiter is a comma.

Enter the ASCII decimal number of the column delimiter or press [F2] to display all ASCII characters. For example, for a column delimiter of comma, enter:

```
44
```

Value delim

A value delimiter specifies the ASCII character used to separate the individual values in a multivalued column. The value delimiter is limited to 1 byte in length. Enter the ASCII decimal number of the value delimiter or press [F2] to display all ASCII characters.

Text marker

A text marker is used to distinguish between a character that is actually data in a column, and a character that is being used as the column delimiter. For example, an ASCII file that uses commas as column delimiters may contain a text column that includes commas. Specifying a text marker ensures that the bond recognizes commas in data as text rather than as delimiters.

A text marker specifies that when data in a column contains the character that is specified as the column delimiter, that data is enclosed by the text marker. The text marker is limited to 1 byte in length. In most cases, the text marker is " (a double quote mark).

Some products create ASCII files in which all columns containing non-numeric data are enclosed in a text marker character. If you are writing to such a file, you can specify that data in designated columns is *always* surrounded by the text marker when it is written to the file. This specification is made on a column-by-column basis, using the softkey [Shift-F3] (Specify Foreign Column Attributes). For more information, see "Specifying Foreign Column Attributes" under the topic "Defining ASCII columns" earlier in this chapter.

Enter the ASCII decimal number of the text marker or press [F2] to see a list of all ASCII characters.

Assigning keys

The ASCII bond automatically creates an implicit row key column at the time that the dictionary file is created. The implicit key column in the dictionary is called RECNO and its data type is INTEGER.

Note You cannot select any other column to be the key to an ASCII file. You cannot create a multipart key for an ASCII file.

Synonyms for RECNO

When you use Paint to define an ASCII file, the prompt that you identify as the key becomes a synonym for the implicit key RECNO and is added to the dictionary. You can also add a synonym for RECNO at the Dictionary window or at the Maketable window.

Defining columns

To create a new ASCII file or access an existing one, you must define all of the data columns in the row structure.

Use the lower half of the Maketable window to enter column definitions. For more information, see "Using existing ASCII files" above. See also the "Managing tables" chapter in the *User's Guide*.

Column position gaps

Every "F" type column is assigned a position in the row. If there are column position gaps in the definition of the file, rows cannot contain data in these positions. For example, if there is no dictionary definition for position 3 in the row, an attempt to write (save) a row that has data in position 3 will fail.

Note You can display column position information at the Maketable window, the Dictionary window, or using the TCL command LISTDICT.

Copying files and rows

You can use the usual commands in Advanced Revelation to copy and move data in the ASCII bond. However, there are restrictions on doing so.

Copying rows

Unless the row structure of both the source and destination file is exactly the same, you should not copy data using the TCL command COPYROW. Any differences in column definitions will prohibit the copy operation or may result in data loss.

Copying files

To copy an entire ASCII file, use the COPYFILE command. As with other file types, you can only copy files within a single environment. This means that you can copy an ASCII file to another ASCII file only.

Modifying files

Changes you make to bonded ASCII files may be affected by the constraints imposed by the ASCII file structure. Both row length and order are important.

Note You cannot use the TCL command REMAKEFILE to convert an ASCII file to another file type.

Caution! If you intend to make updates to a bonded ASCII file, the Advanced Revelation dictionary must accurately describe the existing file structure. If the dictionary is not accurate, you run the risk of overwriting and corrupting existing data in the bonded ASCII file when rows are added or deleted.

Writing to columns

If you inadvertently violate a column definition by writing invalid data to it, one of several results may occur. The results depend on how you have set the degree of acceptable data loss for the column.

Note For more information about potential data loss, see the chapter "Introduction to Environmental Bonding."

If you have set the degree of acceptable data loss to zero (no loss allowed), the write operation will fail. You may see a message informing you of this. If the degree of loss is set to 1 or 2, the write operation may succeed, but you may experience partial or total data loss.

Adding rows

You can add rows to a bonded ASCII file at any time. However, if you add a row out of sequence, past the end of the file, empty rows (fixed-length files) or row delimiters (variable-length files) are appended to the file. This is necessary in order to assign implicit keys correctly, since the keys are derived from the actual order of the rows in the file.

In a fixed-length ASCII file, for example, if there are 10 rows in the file and you add row 100, ninety rows are appended to the file — 89 empty rows padded with spaces, followed by the data row 100.

In a variable-length ASCII file, if there are 10 rows in the file and you add row 100, 89 row delimiters are appended to the file, followed by the data row 100.

For this reason, it is a good idea to assign the sequential counter %SK% to key prompts when creating windows that access and update ASCII files. For more

information about the sequential counter, see the chapter "Creating windows with Paint" in the *User's Guide*.

Deleting rows

Delete rows from ASCII files by using the DELETEROW command.

Deleting variable-length rows

When you delete a row from a variable-length ASCII file, only the data is removed. The row delimiter is left intact. This ensures that the position of the remaining rows is not changed and that the implicit row keys (location in the file) remain the same.

Deleting fixed-length rows

When you delete a row from a fixed-length ASCII file, the data in the row is converted to spaces. This is done to preserve all row positions in the file and maintain the implicit keys. If you attempt to read the deleted row back, the read operation will fail. See "Null Handling" below.

Null handling

In ASCII files, there is no explicit way to indicate a null row. Instead, an empty string or a string of spaces is used to designate a null row or column.

The ASCII bond makes no distinction in character columns between a null value and an empty string. In variable-length files, the value for NULL is an empty string. In fixed-length files, the value for NULL is blanks (spaces).

In a bonded ASCII file, a row of all null columns (no data whatsoever) is treated as though it does not exist. You cannot deliberately save a null row to the file and then read it back.

For instance, suppose you attempt to write out 5 blank spaces to a fixed-length column that is 5 characters long. When you read the column back in, it will come back as null, because the ASCII bond is incapable of distinguishing between blank spaces and null values.

Indexing

You should apply Advanced Revelation indexing (SI.MFS) *only* to ASCII files that will not be accessed or updated by processes outside of the Advanced Revelation environment.

The position of rows in the ASCII file may shift when a foreign process adds or deletes rows. Since the implicit key is derived from the location of the row, a change in position would change the key assigned to the row.

Anything that causes the implicit keys to change will invalidate the indexes. To keep the index current, you will need to rebuild it after every operation that changes row keys.

Networks

Currently, the ASCII bond supports single user access only. The ASCII bond does no row or table locking. Because no row or table locking is performed, ASCII files can be corrupted if two or more users attempt to write to the same file at the same time. For this reason, you should not allow Advanced Revelation users to access ASCII files that are maintained in a multiuser or network environment.

ASCII files that are accessed with the ASCII bond in Advanced Revelation should be isolated in a single user environment. For example, they might be stored on your local hard disk. Alternatively, your network operating system may provide a method to make files non-shareable.

Default foreign column attributes

The following table lists the default ASCII foreign column attributes for each data type. These attributes determine precisely how data is read for use in Advanced Revelation, and written to the ASCII file. The Justification and Length defaults in the table are for fixed-length ASCII files. For variable-length ASCII files, Justification and Length default to null.

Note If you do specify Length for a variable-length ASCII file, ASCII will pad the column to the appropriate length. The default Justification is L.

ASCII Foreign Column Attributes

Type	Output	Input	Justification	Length
CHAR			L	required
VARCHAR			L	255
INTEGER	MD0	MD0	R	10
DECIMAL	MD0,	MD0	R	19
FLOAT		R	20	
DOLLARS	MD2,$	MD2	R	14
BOOLEAN	BY,N	BY,N	L	1
DATE	D4/	(D)	R	10
TIME	MTHS	MT		10
DATETIME	DTD4/^HS	DT	L	21

33. The dBASE III Environmental Bond

Description of the dBASE III bond

The dBASE III environmental bond enables you to use Advanced Revelation as a front-end to your dBASE III or dBASE III Plus environment. This means that new and existing Advanced Revelation applications can access data from dBASE files. Using the dBASE bond, you can even create new tables that will be accessible to dBASE users.

For more information, see "Introduction to Environmental Bonding" in the *User's Guide*.

When you use the dBASE bond, your dBASE files are maintained in their proper format. Using Advanced Revelation, you can still perform the normal dBASE operations of creating, modifying, or deleting dBASE files or records. At the same time, you have the added sophistication provided by Advanced Revelation tools such as Paint and R/BASIC. Changes you make from within Advanced Revelation are reflected for users who access the data from within dBASE.

Installation

The dBASE bond is automatically installed when you install or upgrade to Advanced Revelation 3.0.

Logon options

When you open dBASE data files in Advanced Revelation, there is a potential to exceed the number of file handles that are normally available. This is because all dBASE indexes and Memo fields are maintained as separate files. Since Advanced Revelation enables you to have more than one dBASE file open at a time, the available file handles can be exhausted rapidly.

To eliminate this problem, Advanced Revelation provides an /H (Handles) logon option. The H option lets you increase or decrease the number of file handles that are reserved for use by Advanced Revelation.

Note The /H option is only supported on systems using DOS version 3.3 or greater.

For example, the SYSPROG user might enter:

```
AREV SYSPROG /Hnnn
```

where the slash (/) indicates that multiple logon options may follow, and *nnn* indicates the number of file handles to reserve, from 5 to 255.

Note *nnn* must not exceed the FILES specification in the current CONFIG.SYS file. To ensure that sufficient file handles are available to Advanced Revelation, the FILES specification in your CONFIG.SYS file should be set at 30 or more.

The dBASE III bond environment

The dBASE bond provides an additional menu and window and an additional TCL command as extensions to the standard Advanced Revelation environment.

The dBASE III menu

To display the dBASE III Filing Systems menu, choose the **Utilities-Developer-Advanced-Bonds-dBASE** option from the Development menu.

The Indexing window

To display the Indexing window, choose the single option **Indexing** at the dBASE III menu.

The DBASE command

For more information, see "Indexing" below.

You can use the DBASE command to display either the menu or window listed above or to associate existing dBASE indexes with a file.

Characteristics of the dBASE III bond

The constraints of the dBASE architecture require that you keep a few issues in mind when using the dBASE bond:

Flat-file architecture

Because dBASE is a flat-file database, both records and fields have absolute locations corresponding to their order in the file. It is your responsibility to create the proper sequence of records and fields.

Fixed-length records

The dBASE files are based on a fixed-length record structure. Consequently, the dBASE bond will not allow you to make any changes to a file that might require structural changes. If you need to make such changes, you must make them within the dBASE environment.

Record keys

The dBASE bond uses "implicit" keys to identify records. The keys are implicit because they do not exist in the dBASE file but are instead derived from the location of the record in the file.

Implicit key assignments are necessary for the dBASE bond to ensure compliance with dBASE architecture. The dBASE bond maintains the proper row order using sequential, implicit keys for the rows.

Note You cannot create a multipart key for a dBASE file.

Maximum records

The dBASE environment does not allow the creation of more than one billion records.

Indexing

You can create and use dBASE indexes for a bonded dBASE file. You can also apply Advanced Revelation indexing (SI.MFS) to a bonded dBASE file, although this is not recommended if the file will be accessed from outside of Advanced Revelation.

Using the dBASE III bond

A bonded dBASE file is for the most part no different than any other Advanced Revelation table. You can use all of the same tools and perform all of the standard operations on dBASE files with the exception of indexing.

Special requirements or operations required to use the dBASE bond are discussed below. In all other circumstances, you should assume a bonded dBASE file requires no special treatment.

Accessing dBASE files

For more information on volumes, see "Managing volumes" and "Introduction to Environmental Bonding" in the *User's Guide*.

You must create a volume for your dBASE files before using them. A volume will tell Advanced Revelation that you are using the dBASE "table type", plus the location of your dBASE files. You can create multiple volumes if you have dBASE files in more than one location.

To assign a volume, choose the **DB Admin-Location-Define** options from the Advanced Revelation Development menu or use the SETVOLUME command from the Command window.

Fill in the prompts as for any volume. When indicating a *Table Type* for the dBASE bond, use the name DBASE_BFS. If you do not fill in a value at the *Control Tables Location* prompt, the dBASE bond will place the control tables in a default location (the subdirectory DBASE.BND beneath the current subdirectory).

After you have assigned a volume, you need to attach the volume to make it accessible. Choose the **DB Admin-Tables-Attach** options from the Advanced Revelation Development menu, or use the ATTACHTABLE command at the Command window.

Using existing dBASE files

To access existing dBASE files, create a volume using the SETVOLUME command. You can then access existing dBASE files on that volume by simply attaching them. When you first access an existing dBASE file, an Advanced Revelation dictionary is automatically created for the file. The existing fields in the dBASE file are mapped to corresponding dictionary definitions in Advanced Revelation. In addition, an implicit key column is created in the Advanced Revelation dictionary.

The dictionary created for a table is assigned the standard Advanced Revelation identifier of:

```
DICT.tablename
```

where *tablename* is the name of the existing dBASE file. To display the dictionary, use the TCL command LISTDICT.

Modifying the dictionary

For more information, see "Modifying the file structure" below.

You can add or delete type G (Group) or type S (Symbolic) columns in the Advanced Revelation dictionary at any time. You can also add or delete type F columns, but only if they are synonyms of existing columns. You cannot add or delete new master columns (columns for which the Master definition prompt in the Dictionary Bonding Information window is set to "Yes").

Defining new dBASE files

To create a new dBASE file, choose the **DB Admin-Tables-Define** options from the Advanced Revelation Development menu or use the MAKETABLE command at the Command window.

Assigning keys

When you define a dBASE file, an implicit key column called RECNO is automatically created in the dictionary. At the Paint window, the name you choose as the key column for your table becomes a synonym for the RECNO column and is added to the dictionary.

You can also add a RECNO synonym at the Dictionary window.

Note You cannot create a multipart key for a dBASE file.

Because dBASE uses implicit keys, there is the possibility that the primary key to a row can change. For instance, if a user "packs" a file from within the dBASE environment in order to compress out deleted records, the remaining records can have new keys (positions) assigned to them. The net effect from within Advanced Revelation is that the key column for a row will have changed.

Data typing

There are a few data types provided that are unique to the dBASE bond. These dBASE types reflect their usage in the dBASE environment. You can use these types or the standard Advanced Revelation data types.

The dBASE bond-specific types are:

- DBASE_C (character)
- DBASE_D (date)
- DBASE_N (numeric)
- DBASE_L (logical)
- DBASE_M (memo)

See "Data type maps" below for bonding dictionary information and the standard type correspondences.

If you use the standard Advanced Revelation data types, correspondences will be automatically applied to map to an appropriate type for the dBASE bonding dictionary.

The following are the types provided by the dBASE bond:

DBASE_C	This type of field may be any printable ASCII character. The maximum length is 254 characters.
DBASE_D	The DBASE_D field is an internal representation of the date as maintained by the dBASE environment.

DBASE_N	A DBASE_N field may be an integer or decimal number. The maximum length is 19 digits, where a decimal point and negative sign each count as a digit. The maximum number of decimal places is 15.
DBASE_L	Fields of this type indicate logical true or false. The maximum length is one character. The internal representation as maintained by the dBASE environment is 0 or 1.
DBASE_M	A DBASE_M field may be any printable ASCII character. The maximum length is 5,000 characters of text.

Assigning field attributes

Whenever you define a dBASE field, only two attributes are potentially available: length and decimal places. Their applicability is determined by the data type you choose.

MAKETABLE will prompt you for the attributes when you enter the data type.

Maintaining field position

Because fields have a fixed location in dBASE record structures, you must maintain an ordered sequence of columns in the dictionary. In other words, there must be a column defined for every position in the row.

Note Each column has a *Position* number. You can display this information in the Maketable window, in the Dictionary window, or by using the TCL command LISTDICT.

The bond will not allow you to create a table with missing columns. If you delete a column when defining the dictionary (before the data table exists), you will need to change the position numbers assigned to the remaining columns to reflect the new sequence.

Modifying the dictionary

You can add or delete type G (Group) or type S (Symbolic) columns in the Advanced Revelation dictionary at any time. You can also add or delete type F columns that are synonyms of existing columns.

You can only add or delete new type F master columns (columns for which the Master definition prompt in the Dictionary Bonding Information window is set to "Yes") as long as the data table does not exist. For example, when using Paint to create a table, you can add or delete master columns until you choose to create the dBASE data file.

Once the table has been created, however, you can only change it by deleting and redefining it, or by restructuring it from within dBASE.

Updating dBASE files

Using your dBASE bond, you can update dBASE files as if they were Advanced Revelation tables. At the same time, users of the dBASE environment will see your changes as if they had been made in the dBASE environment.

The following topics address issues of which you should be aware as you update dBASE files.

Writing to columns

Any data you write to a column must conform to the type, length, and possibly decimal places of the column as defined in the dictionary.

For more information about potential data loss, see "Introduction to Environmental Bonding" in the *User's Guide*.

If you inadvertently violate a column definition by writing invalid data to it, one of several results may occur. These depend on how you have set the degree of acceptable data loss for the column.

If you have set the degree of acceptable data loss to zero (no loss allowed), the write operation will fail. You may see a message informing you of this. If the degree of loss is set to 1 or 2, the write operation may succeed (it will not be rejected), but you may experience partial or total data loss.

Adding a row

For more information, see "Deleting a row" below.

You can add rows to a bonded dBASE file at any time. However, if you add a row out of sequence, past the end of the table, rows marked as "deleted" are appended to the table.

For more information about the sequential counter, see "Creating windows using Paint" in the *User's Guide*.

For example, if there are 10 rows in a table and you add row number 100, 89 "deleted" rows are appended to the table. For this reason, it is a good idea to assign the sequential counter %SK% to key prompts when creating dBASE records.

Deleting a row

When you delete a row, the data is removed, but the position in the table for the row is maintained. In order to preserve the dBASE architecture, the row is marked as deleted and its key (location in the table) remains.

If you subsequently write to the deleted row, it is automatically marked as undeleted and the new contents are added.

Copying tables and rows

You can use the usual commands in Advanced Revelation to copy and move data in the dBASE bond. However, there are restrictions on doing so.

Copying rows

In many cases, you can use the COPYROW command to copy individual rows to and from a dBASE file. As a rule, the target environment determines whether you can copy a row to it. In the case of the dBASE bond, the rows you are copying to a dBASE file must match the file structure. In other words, each column of each row must match the dBASE definition for that field (data type, length, etc.). If you attempt to copy a row with invalid data to a dBASE file, the copy operation will fail.

Note If you have set the degree of acceptable data loss for a column to 1 (accept some data loss) or 2 (accept total data loss), the copy operation might be successful even if there is a mismatch. However, you may end up loFor more information about potential data loss, see the chapter "Introduction to Environmental Bonding".

On the other hand, it is relatively easy to copy records from a dBASE file into an Advanced Revelation table, since an Advanced Revelation table supports a much more flexible row structure.

Copying tables

To copy an entire dBASE file, use the COPYTABLE command. As with other file types, you can only copy tables within a single environment. This means that you can copy a dBASE file to another dBASE file only.

Note You may find it easier to copy the dBASE file using an operating system COPY command. You can then attach the copied file from within Advanced Revelation, and the dictionary for it will be created automatically.

Modifying the file structure

If you change the structure of a dBASE file from within dBASE, the Advanced Revelation dictionary for that file no longer accurately represents the structure of that file.

Note Currently, you *cannot* alter the structure of a dBASE file using any Advanced Revelation tool, for example, the TCL command REMAKETABLE.

During an Advanced Revelation session, the dBASE bond checks for changes in a table's structure the first time you execute an Advanced Revelation process that opens the table. At that time, the bond compares the structure of the table with the Advanced Revelation dictionary and ensures that:

- An Advanced Revelation master dictionary row exists for every column in the dBASE file.
- Data type specifications in the Advanced Revelation dictionary and in the dBASE file match.
- Length specifications in the Advanced Revelation dictionary and in the dBASE file match.

If there is a discrepancy, an error message displays. You must delete and re-create the associated Advanced Revelation dictionary before you can access data in the table again.

Re-creating the dictionary

To create a dictionary that reflects the current table structure, follow these steps:

1. If the current dictionary contains symbolic, group, and synonym columns, copy those rows to a temporary table.
2. Delete the current Advanced Revelation dictionary. At the Command window enter:

   ```
   DELETETABLE DICT.tablename
   ```

   Caution! If you do not include "DICT.", the *data* portion of the table is also deleted.
3. Log out of Advanced Revelation and log back in.
4. Reattach the volume. At the Command window enter:

   ```
   ATTACHTABLE volumename
   ```
5. Re-create the dictionary. At the Command window enter:

   ```
   LISTDICT tablename
   ```

   Because no dictionary exists, one is generated automatically. The new dictionary reflects the current structure of the dBASE file.
6. Copy the rows for any symbolic, group, and synonym columns into the new dictionary and check the dictionary to ensure that:
   - All references in symbolic columns are still correct with respect to column positions (for example, @RECORD<1>).
   - The positions of synonym columns still match the positions of the corresponding master columns.

Modifying the dictionary

You can add or delete type G (Group) or type S (Symbolic) columns in the Advanced Revelation dictionary at any time. You can also add or delete type F columns, but only if they are synonyms of existing columns. You cannot add or delete new master columns (columns for which the Master definition prompt in the Dictionary Bonding Information window is set to "Yes").

Indexing

You can use dBASE indexes with your bonded dBASE files. You can also add Advanced Revelation indexes to your bonded dBASE files. This is *not* recommended if there is any chance whatsoever that users will add or delete rows to the table from outside of Advanced Revelation, or if dBASE users might "pack" the table to compress out deleted rows. If any of these operations occurs outside of Advanced Revelation, the Advanced Revelation index will be out of date, and will have to be rebuilt.

Caution! There is no automatic way from within Advanced Revelation to detect whether an index will need to be rebuilt.

When you use a dBASE index using the dBASE bond, you have the following capabilities:

- Associate existing dBASE indexes with a table.
- Create new dBASE indexes.
- Redefine dBASE indexes.
- Disassociate dBASE indexes from a table.

You can associate up to nine indexes with a bonded table. When you associate indexes with a table, the dBASE bond maintains the association between your sessions — you do not need to re-associate the indexes the next time you log into Advanced Revelation.

Both R/LIST and SQL will automatically use any dBASE indexes associated with a bonded table for reduction, as long as the indexed expression is a column name.

Associating existing indexes

In order to associate existing indexes with a bonded table, both the table and the indexes must exist on the same volume.

There are two methods you can use to associate existing dBASE indexes with a bonded dBASE file:

The Indexing window

Access the dBASE Indexing window by choosing the **Utilities-Developer-Advanced-Bonds-DBASE-Indexing** options from the Advanced Revelation Development menu.

The Indexing window consists of three prompts. At each prompt you can press [F2] or click <Options> to display a listing of optional entries for the prompt, or [F1] for online help.

To associate an index, enter:

1. The name of the bonded table at the *File Name* prompt. Do not use a file extension — the extension .DBF is assumed. After entering the file name, any indexes currently associated with the file will be displayed.
2. The name(s) of the dBASE index(es) you wish to associate at the Index Name prompt. You can specify up to nine indexes. Do not use a file extension — the extension .NDX is assumed.
3. The existing field name or expression indexed at the Index Expression prompt

Note Although the dBASE bond will maintain existing complex expressions (such as equations and functions) used at the *Index Expression* prompt, you will not be able to create new indexes using complex expressions.

Press [F9] or click <Save> to associate the index(es).

The DBASE command

The DBASE command enables you to associate existing indexes from the Command window. However, you cannot create new dBASE indexes using the DBASE command.

The syntax for the DBASE command is:

```
DBASE SET INDEX bonded_table TO index
```

where *bonded_table* is the name of the bonded dBASE file, and *index* is the name of the index(es) to associate. The file extensions .DBF and .NDX are assumed, and should not be specified.

One form of the DBASE command displays a menu and the other displays a window:

- The format:

  ```
  DBASE
  ```

  displays the dBASE III menu.

- The format:

  ```
  DBASE SET INDEX TO
  ```

  displays the Indexing window.

Creating indexes

Use the dBASE Indexing window to create a new dBASE index. Display the Indexing window by choosing the **Utilities-Developer-Advanced-Bonds-DBASE-Indexing** options from the Advanced Revelation Development menu, or by using the command format DBASE SET INDEX TO from the Command window.

You may index any of the following types of columns:

- Character columns with a maximum of 100 characters
- Numeric columns
- Date columns

To specify new dBASE indexes:

1. At the *File Name* prompt enter the name of the bonded file. Do not use a file extension — the extension .DBF is assumed. After entering the file name, any indexes currently associated with the file will be displayed.
2. At the *Index Name* prompt enter the name(s) of the dBASE index(es) you wish to create. Do not use a file extension — the extension .NDX is assumed. You can specify up to nine indexes.
3. Finally, at the Index Expression prompt enter the name of the field to index. You cannot specify a complex expression.

Press [F9] or click <Save> to create the index(es).

Redefining indexes

To redefine an index, enter a new field name at the *Index Expression* prompt of the Indexing window. You cannot specify complex expressions. The field names must be acceptable index types (see "Creating indexes" above).

When you change indexed fields, the dBASE bond deletes the current index and rebuilds the index using the new field(s).

Disassociating indexes

To disassociate a dBASE index from a bonded table, delete the appropriate index from the listing displayed at the *Index Name* prompt of the Indexing window.

This does not delete the .NDX file stored in the volume, but only removes the established association.

Networks

When you use the dBASE bond, all record and table locks that you execute will be honored. You will be coordinating not only with other Advanced Revelation users of the table, but with other dBASE users as well.

The dBASE III bond reference

The following information is important for your proper use of the dBASE bond:

Naming conventions

The following conventions apply to all bonded dBASE files:

File names

Although the Advanced Revelation name for your bonded table can be any valid Advanced Revelation identifier, this name may not be a valid file name at the operating system level.

Consequently, when assigning a *foreign* table name for the file, the dBASE bond converts and truncates file names to conform to your operating system requirements.

Note Truncation may cause a conflict with an existing file name at the operating system level.

When converting a file name, invalid characters are changed to the underscore character (_), unless the character is the first letter of the name. In that case, the letter is changed to "A".

File extensions

Data tables use the standard dBASE extension .DBF. You do not need to indicate this as part of the table name — it is appended automatically when you name or access the table.

Index files use the standard dBASE extension .NDX. As with the .DBF extension, you do not need to indicate this, since it is always assumed.

Field names

Field names have a maximum length of 10 characters. Valid names may not contain spaces and must begin with a letter. A name may consist of letters, numbers 0 through 9, and the underscore character (_).

Field names exceeding the allowable length are truncated for assignment to the dBASE file. The original Advanced Revelation field name is maintained for use inside Advanced Revelation.

Data type maps

These tables illustrate the data type requirements for the bond and the correspondences used to convert data types between dBASE and Advanced Revelation environments. The following table lists the dBASE dictionary attributes.

Note You can display or specify this information at the Dictionary Bonding window ([Shift-F5] from the Dictionary window).

dBASE Bonding Dictionary

		(Foreign Field Attributes)		
Type	**Data**	**Length**	**(Default)***	**Decimal**
C	Character	1-254		—
N	Numeric	1-19	(19)	1-15
D	Date	8	(8)	—
M	Memo	10**	(10)	—
L	Logical	1	(1)	—

*The default for Length (the value used if none is provided) is the value in parentheses. Length is required for Character, so there is no default value.

**In the primary file. Occupies up to 5,000 bytes in a secondary file.

The following is the mapping of dBASE to Advanced Revelation data types. These correspondences are used by the bond when automatically constructing dictionaries:

dBASE to Advanced Revelation

Type	Data	Advanced Revelation Type
C	Character	CHAR(length)
N	Numeric	INTEGER(low, high)
N	Numeric w/decimal	DECIMAL(length, decimals)[1]
D	Date	DATE
L	Logical	BOOLEAN
M	Memo	VARCHAR(5000)

[1]The difference between *length* and *decimals* must be 2 or greater.

The following is a mapping of the standard Advanced Revelation data types to the dBASE data types. The bond applies these correspondences if you do not use the data types provided by the bond (see "Defining new dBASE Files"):

Advanced Revelation to dBASE

Type	dBASE type	dBASE Length	dBASE Decimal
CHAR	C	< 255	—
VARCHAR	M	10[1]	—
INTEGER	N	< 20[2]	—
DECIMAL	N	< 20[3]	< 16
DOLLARS	N	< 20[2]	2
BOOLEAN	L	1	—
DATE	D	8	—
TIME	n/s (N)[4]	< 20[2]	—

Notes

[1]In the primary file. Occupies up to 5,000 bytes in a secondary file.

[2]Including negative sign.

[3]Including negative sign and decimal point.

[4]The designation "n/s" indicates that the type is not directly supported in dBASE. Data can be stored in these fields using an alternative type, but cannot be used in dBASE for the purpose suggested by the data type.

Utilities

34. Configuring the workstation

About workstation configurations

Advanced Revelation allows you to customize the way it behaves so you can match it to the type of workstation you have. Among the characteristics that you can change are:

- The colors you see throughout Advanced Revelation.
- The use of a high-density monitor (video display).
- The behavior of your mouse, if you use one.
- The type of printer that you use.
- The use of expanded memory.
- Communications options, if you want to use Advanced Revelation as a terminal emulator.

About workstations

Your workstation is the computer that you work on. If you move to a different computer then you are using a different workstation.

The workstation settings described in this chapter apply to the workstation you are using—they're not related to your user name or application. The advantage of this is that you can change computers and open your application as always, but you will be taking advantage of the capabilities of the computer you are using.

About workstation settings

Advanced Revelation provides default settings for two different types of workstations: a computer with a color monitor and one with a monochrome monitor. Other than the color settings, the configurations are the same.

Two (or more) workstations on your network might be similar or identical. Configurations are therefore centrally located, so all users on a network can share the same configurations. By the same token, though, if any one user on the network changes workstation configuration, those changes will affect all other stations that share that configuration.

The default configurations are fine for the majority of workstations. You might want to fine-tune them to your requirements (example: establish the proper printer or printers), and then simply rely on these default configurations. If you do this, you will not need to create custom workstation configurations at all.

Note Certain situations do require custom configurations. Examine the list under "Creating a custom workstation configuration" below to see if you should create custom configurations.

About configuration files (INI files)

Workstation settings are stored in operating system files. When you log onto Advanced Revelation, it looks for a particular configuration file and loads the settings it finds there. The configuration files always have an extension of INI (example: AREVC.INI).

Advanced Revelation provides two default configuration files: AREVC.INI and AREVM.INI. The AREVC file stores a configuration for a workstation with a color monitor, and AREVM that for a monochrome monitor. If you create a custom configuration, you can copy one of these to create your own INI file.

Changing workstation configurations

There are many settings you can change for your workstation, and different ways of making these changes. The steps listed below are therefore just general steps describing how to get to the places in Advanced Revelation where you can make changes. Individual changes are described later in the chapter.

When you display one of the workstation configuration windows, you will see the settings that are current for your workstation. If you are using a default configuration (example: default color configuration), you will see these. On the other hand, if you have loaded a custom configuration, you will see the settings stored in your custom configuration.

To change the current configuration:

1. Close your current application or log onto Advanced Revelation without a user or application name. The Opening menu displays.
2. Choose **Options-Configuration-Hardware** to see the configuration menu.
3. Choose the type of configuration you want to change (Display, Printer, etc.). The corresponding window displays for the configuration that you have loaded at the time.
4. Make your change and save it with [F9] or by clicking <Save>. The next time you log onto Advanced Revelation your change will take effect.

Changing other INI files

You can change any Advanced Revelation INI file while in the hardware configuration windows. Enter the window as you would normally, then press [Shift-F1] to the see the INI-filename prompt. Press [F2] or click <Options> to see a list of the INI files in the

current path. If you select one of these, the changes you make in the configuration window will be for that INI file only.

Creating a custom workstation configuration

You might want to create a custom workstation configuration file for one of these reasons:

- One or more workstations on your network has different capabilities than the majority of workstations (example: a VGA screen).
- Individual workstations on a network have access to different printers.
- You are using a computer (example: a laptop) that has a monochrome display but uses a color-capable video card.

To create a custom workstation configuration, make a copy of one of the default configuration files and then modify the new one to suit your requirements. Choose the default configuration file that is closest to your desired configuration. For example, if you are setting up a configuration for a computer that has a color monitor, use the default configuration file AREVC.INI as the basis for your changes.

Setting the environment variable

If you are using one of the default configurations, Advanced Revelation automatically recognizes and uses the appropriate INI file. However, in order for Advanced Revelation to recognize that you are using a custom configuration file, you must set an operating system environment variable before you log into Advanced Revelation. We recommend that you make a change to your AUTOEXEC.BAT file in order to set this variable. However, you can set the variable at any time, so long as you do it before you log onto Advanced Revelation.

To create a custom workstation configuration:

1. Quit Advanced Revelation and be sure you are at the subdirectory where you have installed Advanced Revelation.
2. Copy one of the two default INI files (AREVC.INI or AREVM.INI) to a new name. Make sure that the new file also has the extension INI.
3. Add the following statement to your AUTOEXEC.BAT file:

   ```
   SET AREV=x:\path\name.INI
   ```

   where "x:\path\" is the drive and path where the new INI file resides, and "name" is the name of the INI file that you just created. Do not put a space between the word AREV and the equal sign.
4. Re-run your AUTOEXEC.BAT file or reboot your computer in order to load the environment variable with the new configuration name.

5. Log onto Advanced Revelation with no user or application name.
6. Follow the instructions for changing your configuration to select the settings you require. Remember that any changes you make will affect only the new INI file.

Setting colors and border types

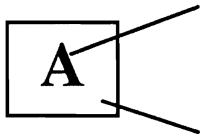
You can set the default colors used in windows, menus, popups, messages, the status line, and the editor window in Advanced Revelation. Of course, you always have the option to override these default colors when you create a window, etc. You can also set the color of the Advanced Revelation "backdrop", the blank area between the menu and the status line when there are no windows or popups active.

About setting colors

When you set colors you choose from a palette of 16 possible colors. These are actually eight basic colors (black, blue, green, cyan, red, magenta, brown, and gray) in combination with the attributes "light" and "dark" to form a total of 16 colors.

If you are using a monochrome display, you set video attributes such as highlight, reverse, and blink. Because of the limitations of monochrome hardware, you cannot get a full choice of 16 attributes.

You can set colors for two areas, the foreground and the background. The foreground is the text and other characters that you see. The background is the field around the character that contrasts with it:



The character is in the foreground.

The background is the contrasting field around the character.

When you set a color, you can set just the foreground, or the foreground and the background together. If you set just the foreground, Advanced Revelation uses the default background (for example, for the particular window you are in). If you set both together, then you get that particular combination regardless of what other color settings may be current in the window, menu, etc.

Depending on how your hardware is set up, setting a background color to a light color may cause the display to blink. Because so many people find it uncomfortable when their entire display blinks, Advanced Revelation prevents you from selecting a light color as a background color. Anytime you choose a background color, Advanced Revelation will display the dark version of that color.

About the color palette

You choose colors from a color palette. Whenever Advanced Revelation prompts you for a color, you can press [F2] or click <Options> to see this palette. At the bottom of the palette you will see the name of the currently selected color.

To choose a color from the palette, use the keyboard or mouse to move to the color you want, then press [Enter] or click twice with the mouse. The name of your color is returned to the prompt.

To toggle between setting foreground colors and setting foreground and background colors, press [F2] or click <Options> while in the color palette. The name at the bottom of the palette will change to reflect your choice of foreground and background (example: "light gray on dark blue" to indicate foreground and background).

To set background colors to light colors, press [PgDn] at the color palette. You will see what the display would look like if you chose light background colors (for example, the palette may blink).

Setting monochrome attributes

If your computer has a monochrome video card installed, the color palette will display the more limited set of monochrome video attributes: normal, highlight, underline, and reverse. These will be repeated across the palette, because different combinations of colors produce the same monochrome video attributes. To see blinking attributes, press [PgDn] at the palette.

About border types

You can also set a border type for most of the entries that are color-sensitive. Border types are identified by a number, and range from double-line borders to borders that have only corners.

Setting window, menu, and popup colors

You can set most system colors in the Windows Video window. From the Opening menu choose **Options-Configuration-Hardware-Video-Windows.**

Depending on what you are setting colors for, you have six to eight choices (border color, title color, label color, etc.). For more information on what each choice is, press [F1] for a description of where your color setting will be used.

To set a border type, at the *Border type* prompt press [F2] or click <Options> for a list of types. Save your changes with [F9] or by clicking <Save>. Your colors will take effect the next time you log onto Advanced Revelation.

Setting status line and backdrop colors

Status line and backdrop window colors are set in their own window. To display this window, choose **Options-Configuration-Hardware-Video-Environment.**

Save your changes with [F9] or by clicking <Save>. Your colors will take effect the next time you log onto Advanced Revelation.

Setting locations for temporary files

Under some circumstances Advanced Revelation creates temporary operating system files. Examples include:

- When sorting large tables
- When searching large indexes
- When displaying an R/LIST or SQL report in a View window
- When printing to a PostScript printer
- When temporarily exiting to the operating system (suspending)

By default, these files are created on the directory where you installed Advanced Revelation. However, you can establish for each workstation where these files will be created.

About creating temporary files

Sort file location

It is important to establish a correct location for these temporary files because you must make sure that they are created on a drive or directory on which there is plenty of room, and to which you have proper file-creation rights. You can also redirect the temporary files to a particularly fast disk (perhaps a local disk) so that these processes run more quickly. Finally, you should direct sort files to directories where there are few or no other files, so that access to the sort file is quick.

Exiting (suspending) to the operating system

When you temporarily exit (suspend) to the operating system, Advanced Revelation stores its current memory configuration into a temporary table (a "rollout" file) of about 500K. If you have a RAM disk, you can make this process happen almost instantaneously.

Note You can always establish a temporary rollout file location for one session using the TCL SETROLLOUT command.

Establishing sort file location

Please read "About creating temporary files", above. To establish a sort file location:

1. From the Opening menu, choose **Options-Configuration-Hardware-Workstation**. The Workstation Hardware window displays.
2. At the *Sort DOS path name* prompt, enter a full path, including the drive letter if necessary. Do not enter a file name, just a path. *The directory you name must already exist.*

   Note If you are changing a workstation configuration that is shared by several stations, be sure to use a drive designation that is available to all of those stations (for example, avoid drive D: if not all stations have such a drive).
3. Press [F9] or click <Save> to save your changes. Your changes take effect the next time you log onto Advanced Revelation.

Establishing rollout file location

Please read "About creating temporary files", above. To create a rollout file location:

1. From the Opening menu, choose **Options-Configuration-Hardware-Workstation**. The Workstation Hardware window displays.
2. At the *Rollout DOS filename* prompt, enter the file name for the rollout file, including a full drive and path. Make sure that the filename is unique for each separate workstation. *The path you enter must already exist.*

   Note If you are changing a workstation configuration that is shared by several stations, be sure to use a drive designation that is available to all of those stations (for example, avoid drive D: if not all stations have such a drive).
3. Press [F9] or click <Save> to save your changes. Your changes take effect the next time you log onto Advanced Revelation.

Listing system configuration information

To see a listing of the system information current for your workstation and session, choose the option **Help-System Info** from the Opening or the Development menus. A read-only window will display, listing information about your current serial number, user and application names, use of memory, network type, use of expanded memory, and more. You can also access this window by using the TCL WHO command.

Using high-density display screens

Advanced Revelation supports high-density EGA and VGA screens. In addition to the standard 25 line by 80 character screen, you have the following options for screen height (all screens are 80 characters wide):

EGA	43
VGA	43 or 50

Changing display types

You must exercise some caution when changing video display modes:

- Advanced Revelation will warn you, but will not prevent you from choosing a display mode that isn't supported by your current video card. If you choose an incompatible display mode, you may not be able to log onto Advanced Revelation properly, or you may find the display defaulting to 25 by 80.
- Conversely, using higher density video display modes, you can create a window that is 43 or 50 lines deep. If this window is then displayed on a 25 line screen, the results are unpredictable. To ensure accurate display on all screens, do your window development on a standard 25 line monitor.

To change video display modes:

1. From the Opening menu, choose **Options-Configuration-Hardware-Display-Environment**. The Video display parameters window displays.
2. At the *Video display mode* prompt, press [F2] or click <Options> to see a list of types. Choose the video type you want to use.
3. The values at *Screen height* and *Screen width* will change to match the mode you selected. You can override the value there, though you cannot exceed 80 columns for width.
4. Save your changes with [F9] or by clicking <Save>. The next time you log onto Advanced Revelation your change will be active.

Using a mouse with Advanced Revelation

You can use a mouse or other compatible pointing device to move the cursor in data entry windows, the Editor, and Paint, and to move the highlight and select options in menus and popups. Advanced Revelation detects automatically if you have a mouse driver installed on your computer and produces a mouse pointer on the screen if the mouse driver is active.

You can adjust the speed sensitivity of the mouse. To do so, choose **Options-Configuration-Hardware-Workstation** from the Opening menu. At the *Speed sensitivity* prompt enter a number from 1 to 100. The higher the number, the faster the mouse pointer moves. Your changes are effective as soon as you leave the prompt. Save your changes with [F9] or by clicking <Save>.

Using printers with Advanced Revelation

Advanced Revelation enables you to select up to three printers for each environment. From these three printers, you can then choose an active printer, which is the printer that the system directs all printer output to. You can switch to another active printer at any time.

Advanced Revelation supports most popular printers, including PostScript printers. However, if your printer does not appear in the list of supported printers, you have these options:

- Check your printer documentation to see if your printer can emulate one of the supported printers.
- Create a printer definition for your printer. See "Creating printer definitions" later in this chapter.

Selecting printers

Before selecting printers, you should know:

- What type of printer you have (or what type it emulates).
- The computer port or ports to which you want to send output. By default, Advanced Revelation sends output to LPT1 (operating system default).
- The height and width in rows and columns, respectively, of a default page on your printer. If output is longer than the default page height, the system starts a new page. If output is wider than default page width, text wraps onto the next line. You can override these values when you activate a printer.

To select printers for your system:

1. From the Opening menu, choose **Options-Configuration-Hardware-Printer-Environment**. The Printer environment window displays.
2. Press [F2] or click <Options> at the *Printer type* prompt to see a list of defined printers, then choose your printer.
3. At the *Port* prompt, select the port to which you want printer output sent.
4. Repeat steps 2 and 3 for up to three printers.

5. At the *Default page height* and *Default page width* prompts, enter the default page size values that you want to use.

Selecting the active printer

After you have selected the printers for your environment, you can select the active printer. This is the printer that all printer output is directed to until you change to another active printer.

Choose **Options-Configuration-Hardware-Printer-Select** from the Opening menu. From the popup, choose the printer you want to activate. Advanced Revelation then initializes the printer (if that's part of the definition for that printer).

Controlling printer output

After you have activated a printer, all output from Advanced Revelation will appear on that printer. You also then have the possibility to control more directly the different attributes that your printer is capable of, such as character formatting (bold, italic, etc.), specific fonts, initialization, and pre- and post-job processing.

To take full advantage of the character formatting capabilities of your printer (bold, italic, fonts, etc.) you must:

- Create Merge reports
- Program reports in R/BASIC

However, you can add form-feed and pre-and post-job processes to any printer definition.

Setting up form-feed control

Laser printers often keep the last page of a print job in the printer until a form-feed command is issued. You can have Advanced Revelation issue a form feed at the end of each print job automatically. To do so:

1. From the Opening menu, choose **Options-Configuration-Hardware-Printer-Definitions**. The Printer definitions window displays.
2. At the *Printer* prompt, enter the name of your printer. Press [F2] or click <Options> to see a list of printers.
3. Move to the third page of the window. At the *Force form feed at end of job?* prompt, enter "Y". Save your change with [F9] or by clicking <Save>. The change will take effect the next time you log onto Advanced Revelation.

Setting pre- and post-print processes

You may need to run network or printer-specific processes before you can print Advanced Revelation reports. For example, you might need to issue a network-specific "capture" command before printing to a network printer, and an "end of job" command to tell the network that you are through printing.

There are two types of printer processes:

- **Open**. Use the Open printer process to establish a connection to a network printer, to send PostScript printer output to a spool file, or to execute other device-specific open processes.
- **Close**. Use the Close printer process to close the connection to a network printer, to print PostScript printer output from a spool file, or to execute other device-specific close processes.

The Open command is run for the default printer when you first log onto Advanced Revelation, or when you select a new printer. The Close command is run to close out the old printer when you select a new one, and when you log out of Advanced Revelation.

You can also run both open and close processes using the TCL PRINTPROC command.

Printer processes are defined using codes and commands. In most cases, these will be:

X To run an operating system-level command such as the NetWare CAPTURE command.

S To run an R/BASIC subroutine that you have written.

For example, you might want to run the NetWare CAPTURE command when you initialize or change printers. You could then issue an "end-of-job" command by re-selecting your printer, which will cause Advanced Revelation to execute the post-print process and run the CAPTURE command. In that case, the code for both processes would be "X" and the command portion would be:

```
PC EXIT CAPTURE /options
```

where "options" would be the correct options for your version of the CAPTURE command.

To establish a printer process:

1. From the Opening menu, choose **Options-Configuration-Hardware-Printer-Definitions**. The Printer definitions window displays.
2. At the *Printer* prompt, enter the name of your printer. Press [F2] or click <Options> to see a list of printers.
3. Move to the third page of the window. At the *Open* prompt, enter the code and command for the Open printer process. At the *Close* prompt, enter the code and

command for the Close printer process. Save your changes with [F9] or by clicking <Save>. Your changes will take effect the next time you log onto Advanced Revelation.

Using PostScript printers

In most cases, printing reports from Advanced Revelation is completely automated. You run the report and send it to a printer, and the system takes care of the rest.

PostScript printers are a little different from most printers, because they process output a page at a time rather than a line at a time. To compensate for this difference, Advanced Revelation sends all PostScript printer output to a spool file instead of sending it straight to the printer. When you print the contents of the spool file, the system formats this line-oriented output into pages for the printer.

Printing on a PostScript printer

To print on a PostScript printer you must first open a spool file so Advanced Revelation can save data there. There are different ways to do this:

- When you log on, if the default printer is a PostScript printer, the system automatically opens the file.
- If you select PostScript as your printer sometime during your Advanced Revelation session, the system opens the spool file for you.
- Use the TCL PRINTPROC command.

Caution! While the spool file is open, do not redirect printer output to a file (using the TCL PDISK command), or you may lose the data in the PostScript printer spool file.

Once the spool file is open, you can run as many reports as you want. Until you close the spool file, Advanced Revelation continues to send output to that file. The size of the file is limited only by the amount of available disk space.

When you want to start sending output to the printer, close the spool file. Again, there are several ways to do this:

- Re-select the PostScript printer as the active printer.
- Execute the TCL PRINTPROC command.

Related topics

TCL PRINTPROC and SETPRINTER commands.

Deleting reports from the spool file

If you want to delete all reports in the spool file, run the TCL command:

```
PRINTPROC OPEN
```

This produces a message that asks if you want to overwrite the spool file. Answer "Y", and:

- The contents of the spool file are deleted.
- The spool file is opened again and ready to receive output.

Caution! Executing the PRINTPROC OPEN command deletes all reports that you have run since opening the spool file, not just the most recent report.

Specifying a location for the spool file

Advanced Revelation will put the spool file for PostScript output in the same place it puts temporary sort files. Therefore you will want to avoid a location that has only limited space, because you may run out of room when printing large print jobs.

Also be aware that by changing this location, you are changing it for everyone who shares your workstation configuration. For this reason you will usually want to avoid choosing a local drive.

To change this location:

1. From the Opening menu, choose **Options-Configuration-Hardware-Workstation**. The Workstation hardware window displays.
2. At the *Sort DOS path name* prompt, enter a complete filename, including drive, path, and filename. Save your change with [F9] or by clicking <Save>.

Logging off while a spool file is open

If you attempt to log off before printing the reports in the spool file, Advanced Revelation prompts you to either print or delete the contents of the file.

Creating printer definitions

If Advanced Revelation does not include a printer definition compatible with your printer, you may want to create a new printer definition.

About printer definitions

A printer definition is a table that translates print attributes such as "bold", "italic", and similar printer commands into the "language" understood by your printer. In addition, a printer definition allows you to define these styles as display attributes for

screen output. For example, you can have bold text display red on your monitor, italic text as green, etc.

Before you create a printer definition, you must:

- Be familiar with the codes and characters that your printer uses to control attributes such as bold, italic, fonts, etc. This type of information is usually available in the manual that accompanies your printer, often in a section that addresses programming.
- Be familiar with an "initialize" or "reset" code for your printer, if there is one. This code resets your printer to its default state.
- Know what fonts and print styles your printer can produce. These might be font names, such as "Courier" and "Times", or styles, such as "Pica" and "Elite", or they could also just be descriptions, such as "letter-quality" and "draft". You will also need to know the codes that your printer uses to produce these fonts.

Entering character strings in the Printer Definition window

As part of the printer definition you will be entering strings of characters to define such attributes as "bold". Enter the strings in one of these formats:

- ASCII. Enter the actual character enclosed in quotation marks.

  Caution! Do not try to enter ASCII characters that have a decimal value lower than 33. These characters have no display equivalent and may cause corruption of the printer definition record. Instead, enter these characters in either their decimal or hexadecimal equivalents.

- Decimal. Enter the decimal values of characters, separated by spaces.
- Hexadecimal. Enter the hexadecimal values of the characters. Precede each value with a "\" (backslash), and separate values with a space.

For example, if your printer requires that you send it the string "$a5L" before it will print in a bold font, you can enter any of the following values:

Character type	Value
ASCII	"$a5L"
Decimal	36 97 53 76
Hexadecimal	\24 \61 \35 \4C
Mixed	'$' 97 "5" \4C

To create a new printer definition:

1. From the Opening menu, choose **Options-Configuration-Hardware-Printer-Definitions**. The Printer definition window displays.

2. At the *Printer* prompt, enter a name of your printer. You can use any name, but we suggest that you use a name that helps you remember the printer type (for example, one printer definition is called HP_LASERJET).
3. At the *Page Height* and *Page Width* prompts, enter the maximum page dimensions in lines and characters, respectively. If output is longer than Page Height, the system starts a new page. If output is wider than Page Width, text wraps onto the next line.
4. For each prompt in the "Begin" column, enter the character string that makes your printer print in the corresponding print style.
5. In the "End" column, enter the character string that causes your printer to stop printing in that print style.
6. At the *Initialize* prompt enter the character string used to reset your printer.
7. Skip the Prelude and End of Job prompts, which are for PostScript printers only.
8. Enter a description in the "Font Name" column to identify which fonts you have associated with which font numbers.
9. For each prompt in the "Begin" column, enter the character string that causes your printer to print in the corresponding font.
10. In the "End" column, enter the character string that causes your printer to stop printing in that font.
11. At the *Open* and *Close* prompts, enter codes and commands for processes that occur when you select and deselect the printer. See "Setting pre- and post-print processes" above.
12. If your printer keeps pages in the printer until the next job starts (as many laser printers do), enter a "Y" at the *Force form feed at end of job prompt.*

Using expanded memory support

Advanced Revelation supports the use of expanded memory software conforming to LIM EMS 4.0. This chapter provides information on how to configure your system and how to invoke expanded memory support.

Note Expanded memory is not necessary if you are using Advanced Revelation for OS/2.

About expanded memory with Advanced Revelation

If your system is configured correctly for expanded memory, Advanced Revelation can take advantage of this resource. Advanced Revelation's expanded memory support

provides additional workspace in RAM for applications. Programmers can take advantage of expanded memory to maintain a greater number of large character strings and large subroutines. This capability is especially important in sophisticated applications that use many supporting routines, large records, or networked applications that have less memory available.

Expanded memory support does *not* allow you to create character strings larger than 64K. Expanded memory also does *not* increase the maximum number of variables, nor does it increase the maximum number of programs. You will still need to exercise caution when programming dimensioned arrays and labeled common areas.

Advanced Revelation's use of memory

Advanced Revelation will use all available conventional memory (conventional memory is the area of RAM up to 640K). If you are using expanded memory manager (EMM) software, Advanced Revelation can also use expanded memory.

If you have an EMM active when you log onto Advanced Revelation, the system will automatically allocate half of the system's available expanded memory for its use, up to a maximum of 4 megabytes. You can use a logon option to increase or decrease the amount of expanded memory that is initially allocated. For more information, see "Logging on with expanded memory" later in this chapter.

Note to C and assembly language programmers

If you have written C and assembly language routines, these can have, at most, two valid pointers into Advanced Revelation string space. The two most recently fetched pointers are guaranteed to be valid. Using more than two pointers will yield unpredictable results.

Caution! Do not use Expanded mode unless you are sure that the C and assembly language routines executed by your Advanced Revelation application have no more than two pointers into Advanced Revelation string space at any given time.

System requirements

To take advantage of expanded memory, you will need:

- A minimum of 256K expanded memory. This can either be as hardware, or by having your EMM emulate extended memory as expanded memory.
- Expanded memory manager (EMM) software that conforms to the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS) version 4.0. Examples include QEMM and 386^{MAX}.
- At least 256K free expanded memory available when you start Advanced Revelation.

Expanded memory support is optimized for use with 386-based machines.

Logging on with expanded memory

Logon options enable you to control how your expanded memory support is used in Advanced Revelation. Available logon options and logon command syntax are explained below.

Advanced Revelation can use expanded memory in three different modes. The mode is determined by the way in which you log on to Advanced Revelation.

Programs-only. (Default mode). Programs longer than 508 bytes of object code are swapped out to expanded memory as necessary. (This does not mean that you can load more programs, only that you can load larger programs.)

Expanded. Any string of characters longer than 508 bytes, including programs, will be swapped out to expanded memory as necessary. Expanded mode is faster than programs-only mode and decreases the chance of running out of conventional memory.

Off. Expanded memory is turned off. Used when conventional memory limits make using expanded memory inefficient and for debugging purposes.

Controlling initial expanded memory allocation

By default, Advanced Revelation uses up to half of the system's available expanded memory, up to a maximum of 4 megabytes. You can increase or decrease the amount of expanded memory that is initially allocated.

In order to use expanded memory at all, Advanced Revelation requires at least 256K of free expanded memory at logon. If your M option specifies less than 256K, Advanced Revelation will allocate at least 256K of expanded memory. If you request more than 4 megabytes of memory, Advanced Revelation will allocate all available expanded memory up to 4MB.

Note If you display the System info window, you may find that the *EM Allocated* prompt displays a slightly lower number than the one that you requested. This is because expanded memory is allocated in 64K increments.

Logon options

To log on with an option for expanded memory, use a switch at the operating system command line:

```
AREV [application] /switch
```

where *application* is an optional application name. *Switch* is the option and can be:

/X	Expanded mode
/O	Off mode
/Mnnn	Request *nnnn* KB of expanded memory (4096=4MB)

Managing expanded memory

To support expanded memory, Advanced Revelation requires three blocks of memory:

- One 16K expanded memory window. Because systems such as Advanced Revelation cannot access expanded memory directly, they require the use of "mappable" regions that they can access. Mappable regions are controlled by the expanded memory manager and are made available to applications for use as windows into expanded memory.
- Two 64K overflow buffers. The overflow buffers are used to prevent data in programs from shifting in memory during critical operations. These buffers may be allocated in mappable regions above 640K if enough mappable memory is available.

Note To determine where Advanced Revelation has allocated the 16K expanded memory window and 64K overflow buffers, use the System info window. For more information, see "Displaying the status of expanded memory" later in this chapter.

Performance is significantly better when one, two, or all three of these blocks are allocated in mappable regions, also known as swapping regions, residing above 640K and below 1M (non-conventional memory).

If mappable regions are not found for the 64K overflow buffers, the buffers will be allocated in conventional memory, and the 16K expanded memory window will be allocated in the page frame. The page frame is a 64K mappable region provided by the expanded memory manager and used to access expanded memory. This not only wastes 48K of the page frame, it also has the undesirable side effect of reducing conventional memory by as much as 128K and possibly reducing the number of variables that can be used.

Therefore, if your expanded memory manager is capable of supporting additional mappable regions, you can improve performance significantly by making as much mappable memory available as possible.

Note If your expanded memory manager supports only one 64K mappable region, namely the 64K page frame, it will be impossible to increase the number of mappable regions.

To make more mappable memory available for Advanced Revelation to use, avoid filling non-conventional memory above 768K. Some expanded memory managers provide the capability of filling unused portions of non-conventional memory with usable high RAM (also called high DOS memory). However, Advanced Revelation cannot directly take advantage of high RAM, so you may want to convert some or all of the high RAM to mappable memory. See your EMM software documentation for details.

Some expanded memory managers may not recognize that areas of non-conventional memory are being used by other peripherals (network cards, for example). When there is a conflict between your EMM and a peripheral device over available non-conventional memory, you may be unable to log on to Advanced Revelation. Your EMM software documentation should provide information about how to detect and resolve this problem.

Displaying the status of expanded memory

The System info window provides information about Advanced Revelation's use of memory. To access this window, choose **Help-System info** from the Opening menu, or use the TCL WHO command. The window is for information only—you cannot make changes at this window.

On the first page of the window, note the information at the *Available memory* and *Expanded Memory* prompts. From the first page of the window, press [PgDn] twice to display page three, which is labeled "EXPANDED MEMORY INFORMATION". When EMM software is active, information displays at the *EM Used*, *EM Allocated*, *16K EM Window*, *64K Overflow Buffer 1* and *64K Overflow Buffer 2* prompts:

Prompt	Displays
Available memory	The number of bytes currently available to Advanced Revelation in conventional memory.
Expanded Memory	"Active" if Advanced Revelation recognizes and is using EMM software. "Inactive" otherwise.
EM Used	The number of bytes in expanded memory that Advanced Revelation is currently using.
EM Allocated	The amount of memory in expanded memory that is currently allocated.
16K EM Window	The starting segment address of the 16K mapped segment.
64K Overflow Buffers (2 prompts)	The starting segment addresses for the two 64K overflow buffers.

Sample configuration for a 386-based system

The EMS software listed below conforms to the requirements for Advanced Revelation:

- 386^{MAX} software-emulated expanded memory from Qualitas, Inc.
- Quarterdeck Expanded Memory Manager software-emulated expanded memory from Quarterdeck Office Systems

This section describes how to create an efficient memory configuration for Advanced Revelation using the Quarterdeck Expanded Memory Manager 386 (QEMM).

Note These instructions do not attempt to replace your expanded memory software or hardware documentation. In fact, it is assumed that you have access to your system's documentation and that you are familiar with its terms and configuration procedures.

To follow this suggested setup, you must have:

- 1MB of extended memory
- Quarterdeck Expanded Memory Manager 386 (QEMM) installed on a 386-based computer

Installing QEMM

Install QEMM, following the documentation provided by Quarterdeck. When using QEMM command line options, do not use the RAM option.

Checking for EMM recognition

Verify that Advanced Revelation can recognize and use the expanded memory manager by checking in the System info window. If Advanced Revelation recognizes and is using the expanded memory manager, the word "Active" will display at the *Expanded Memory* prompt.

Troubleshooting

If you could not log in to Advanced Revelation, QEMM may not recognize that areas of non-conventional memory are being used by other peripherals (network cards, for example). Refer to your QEMM documentation for information about how to eliminate conflicts over available non-conventional memory between QEMM and peripheral devices.

If the *Expanded Memory* prompt does not display "Active", check the following:

- Reboot your system and watch the screen for any error messages indicating that the expanded memory manager is not installed correctly. For example, you may see a message stating that there are errors in the CONFIG.SYS file.
- Verify that you are running a version of QEMM that supports the LIM EMS 4.0 specification.
- QEMM must provide at least 256K of available expanded memory. An error in your setup may reduce the amount of available expanded memory. When this is the case, Advanced Revelation cannot recognize or use the expanded memory capabilities.

Examining your current configuration

After ensuring that Advanced Revelation is using expanded memory, check the segment addresses (shown in hex) displayed in the System info window next to the prompts *16K EM Window*, *64K Overflow Buffer 1*, and *64K Overflow Buffer 2*. For maximum efficiency, all three addresses should be greater than or equal to C000.

Fine tuning your configuration

If you have an EGA or VGA adapter, you may be able to improve your existing configuration. Since Advanced Revelation does not use these modes, memory from A000-AFFF can be used for conventional memory, providing Advanced Revelation with an additional 64K of memory.

Run the VIDRAM utility with the ON switch to make the VGA graphics region of memory available to Advanced Revelation. After your session, run VIDRAM OFF to recover the graphics functions. This procedure allows Advanced Revelation to access more than 700K.

Once you are familiar with Advanced Revelation's use of non-conventional memory, you may be able to make even more memory available by experimenting with the RAM option. Use the RAM option to convert into high RAM any regions in non-conventional memory that Advanced Revelation is not using. Then use the LOADHI.COM program supplied by Quarterdeck to move TSR (terminate and stay resident) programs and device drivers out of conventional memory and into the regions that you have defined as high RAM.

Suggested configuration for a 286-based system

Advanced Revelation will perform best on a 286-based computer that uses EMS software that supports at least 5 pages for mapping. Software emulators of expanded memory for 286-based machines are generally limited to one 64K page frame (which is only 4 mappable pages). Hardware expanded memory usually provides a minimum of 5 mappable pages, therefore hardware expanded memory is likely to produce a more efficient system when used with Advanced Revelation.

The expanded memory managers listed below conform to the requirements of Advanced Revelation:

- XMA2EMS running DOS 4.0 software-emulated expanded memory from IBM
- ABOVE BOARD hardware expanded memory from Intel
- Rampage hardware expanded memory from AST Research, Inc.

Setting communication options for terminal emulation

You can use Advanced Revelation as a serial terminal for larger systems. To do so, you use the TCL TERM command to enter terminal emulation mode.

You can set the communications options (baud rate, parity, etc.) in the command window, or you can store them permanently in a configuration file. To set them permanently, choose **Options-Configuration-Hardware-Communications**. Press [F1] at any prompt for more information about the values you can enter for each communication setting.

35. Using the editor

About the Advanced Revelation editor

Advanced Revelation's editor to enter or edit rows in Advanced Revelation tables. For example, you can use the Editor to edit R/BASIC source code rows. You can also use Advanced Revelation's Editor to enter or edit operating system files.

Accessing the editor

Using the Edit window

1. Choose the option **Define-Program-Edit** from the Development menu.
2. At the *Table name* prompt in the Edit window, enter the name of the table you wish to edit. To display a list of available tables, press [F2].
3. At the *Row(s)* prompt, enter the list of rows you wish to edit, Press [F2] to display all of the rows in the table. You can also enter * (asterisk) to edit all rows in a table.
4. Press [F9].

Entering the EDIT command at the Command window

You can also start the editor by using the EDIT command at the Command window. The general format for the EDIT command is:

```
EDIT {DICT} tablename row.key(s)
```

Changing the edit environment

For the Editor environment, you can alter:

- Tab stops
- These edit mode ([Ctrl-W]):
  - Case sensitivity of searches
  - Automatic wrapping of long text lines
  - Automatic indents for new lines
  - Automatic new line insertion when [Enter] is pressed
- Whether the cursor position and editor status are saved between sessions

Permanently changing the editing environment

To change default settings for a user or application editing environment, choose from the Opening menu the **Options-Configuration-Environment-Editor** option.

Temporarily changing the editing environment

You can temporarily override some Editor environment defaults for the current session.

Changing edit modes

Press [Ctrl-W] to display current edit mode settings. Use the popup to toggle each edit mode.

Changing Tab Stops

Press [Ctrl-T] to display current tab stops. To set or remove a tab without disturbing the current position of other defined tabs, the Editor should be in overwrite mode. If INS displays on the status line, press [Ins].

About buffers

For more information, see "Active Keys in the Editor" later in this chapter.

Advanced Revelation provides five buffers, numbered 1 through 5, where you can store defined blocks of text stored for later use. A buffer acts like a clipboard which holds a segment of a column or row. Because you have five buffers to work with, you can hold up to five different blocks at one time.

Buffers are system-wide and stay active between processes. For example, you can store a block of text from one row in a buffer for use later in another row. The buffers are cleared when you exit Advanced Revelation.

About value windows

When the editor displays a row, each column appears on its own line. Multiple values, sub-values, or lines in a text column are separated by these markers:

Multiple values in a multivalue column	ASCII 253
Sub-values	ASCII 252
Text lines	ASCII 251

To display and edit multiple values, sub-values, or text lines, each on its own line:

1. Move the cursor to the line that contains the values you want to edit.
2. Press [Ctrl-E].
3. After making your changes, press [F9] or click <Save>, and then [Esc] to return back to the row.

Subdividing a column

Each time you press [Ctrl-E], Advanced Revelation further subdivides the current value and displays the subdivision in a new window. Edit the value in the window as you wish. To retain your changes, you must press [F9] at each window as you return through the Value windows to the editor.

Reformatting text

You can reformat a block of text using [Ctrl-Z] (Reformat) while editing. The text in the marked block will be reformatted to fit into the current window width. If you want the text to be narrower or wider than the current window width, use [Ctrl-F7] (Resize) before reformatting to make the window the appropriate width.

Caution! If you press [Ctrl-Z] without marking a block of text first, the entire contents of the window will be reformatted.

About editing operating system files

Accessing operating system files

To edit an operating system file in the editor, use the EDIT command in the Command window with the word DOS as the tablename and the operating system file name as the key:

```
EDIT DOS pathname\osfilename
```

For example, to edit an operating system file named MYNOTES.DOC in a directory named DOCUMENTS, enter this command at TCL:

```
EDIT DOS DOCUMENTS\MYNOTES.DOC
```

Advanced Revelation starts the editor and displays the file in the Editor window.

Note The maximum size of a row you can edit in the editor is 65,536 characters.

Displaying operating system files in the Editor

To display an operating system file properly, use [Shift-F7]. Advanced Revelation displays the table so you can edit it.

Converting to operating system or Advanced Revelation format

When Advanced Revelation converts an operating system table to Advanced Revelation format, the carriage returns and line feeds in the table are replaced by Advanced Revelation column markers, ASCII character 254. In the same way, when Advanced Revelation converts an Advanced Revelation formatted table to operating system format, the column markers are replaced by carriage returns and line feeds.

To convert a table's format, press [Shift-F8]. Use this key as a toggle to switch from operating system format to Advanced Revelation format and back again.

Programming tools in the Editor

Because it is used often to create R/BASIC programs, the editor incorporates tools to help you program. For more information on the topics discussed here, see the *R/BASIC* manual.

Saving and compiling programs

To save and compile a program, press [Shift-F9]. When you press [Shift-F9], Advanced Revelation saves the current row and immediately starts the R/BASIC compiler.

Note The maximum size of source code that can be compiled is approximately 34,000 bytes.

If the compile is unsuccessful, Advanced Revelation returns you to the Editor with the cursor on the first line that contains an error. The Status line displays the error message. Press [Shift-F10] to move to the next error.

Note If the system displays an "out of memory error message", close the Editor, return to the Development menu and choose the **Define-Program-Edit** option.

Editing $INSERT Rows

A $INSERT statement directs the compiler to insert R/BASIC source code from another program into the current program. To edit the program named in an $INSERT statement:

1. Move the cursor to the line containing the $INSERT statement
2. Press [Shift-F2].

Advanced Revelation displays the program so that you can edit it. When you are finished editing the program, press [F9] to save your changes and [Esc] to return to your original program.

Displaying an ASCII chart

You can display a chart of ASCII character codes while you are in the Editor.

1. Press [Ctrl-S] to display the Special Modes popup.
2. Choose Option 5 "ASCII" table.

Advanced Revelation displays the table. Press [F9] to return the character, or [Esc] to leave.

Editing a list of rows

For more information, see "Using windows for queries" in the *User's Guide*.

To edit multiple rows in a browse list:

- Enter an explicit list of keys at the *Row(s)* prompt in the Edit window.
- Enter an * (asterisk) at the *Row(s)* prompt in the Edit window.
- At the Command window, activate a select list. Then enter the EDIT command without any row keys.

If you are editing in a browse list, you can move forward and backward in the list using [Alt-F] and [Alt-B].

Active keys in the Editor

Many of the active keys in the Editor are the same keys that are active in any window.

Help	
Display context help	[F1]
Display general help	[Ctrl-F1]
Display browse list	[F2]
Display concept help	[Ctrl-F2]
Display available softkeys	[F6]
Display available menu	[F10]

Editing, saving, and escaping	
Exit window without saving changes	[Esc]
Save current entry	[F9]
Undo current line changes	[F4]
Restart edit, undoing all changes since last save	[Ctrl-Q]
Refresh/clear window	[F8]

Finding and replacing	
Find string	[Ctrl-F]
Replace string	[Ctrl-R]
Repeat last find	[Ctrl-A]

Deleting and inserting	
Delete character to the left of the cursor	[Backspace]
Toggle insert/overwrite	[Ins]
Delete current character	[Del]
Clear all characters in current line	[Ctrl-X]
Delete characters to end of line	[Ctrl-L]
Delete characters from beginning to current position	[Ctrl-K]
Insert blank line	[Ctrl-N]
Delete current line	[Ctrl-D]
Delete word right	[Ctrl-Y]
Toggle auto insert mode (Popup)	[Ctrl-W]
Toggle auto line indent (Popup)	[Ctrl-W]
Toggle auto line insert (Popup)	[Ctrl-W]
Delete current row	[Alt-D]

Using blocks

Define block toggle (beginning/ending)	[Ctrl-B]
Clear block definition	[Ctrl-U]
Cut block	[Ctrl-F3]
Paste block	[Ctrl-F4]

Merge and break lines

Cut line into two lines at current cursor position	Ctrl-C
Join current line with following line	Ctrl-J

Using row key lists

Display filter key popup	[Ctrl-F10]
Edit browse keys list	[Alt-I]
Next key forward	[Alt-F]
Next key backward	[Alt-B]

Other edit keys

Toggle auto word wrap (Popup)	[Ctrl-W]
Go to line number (user entered)	[Ctrl-G]
Display special modes	[Ctrl-S]
Duplicate last character	[Ctrl-P]
Set tabs	[Ctrl-T]
Edit value window	[Ctrl-E]
Reformat text to fit window	[Ctrl-Z]
Toggle the mnemonic keystroke emulator	[Ctrl-Dash]

36. Importing data into Advanced Revelation

About importing files

Advanced Revelation has utilities that allow you to import three types of files:

- ASCII files, both variable length and fixed length
- dBASE III files (.DBF)
- Lotus 1-2-3 spreadsheets (.WKS)

To import a file, you define an import process. For each process that you define, you provide all the specifics for the import—what file to import and to which Advanced Revelation table, what type of file to import, etc. When you are done defining, you can use your process definition to import the data right away. But the process definition stays on file so you can recall it and re-import the file another time.

Each import process has a unique name. When you assign a name to a process, Advanced Revelation tags the name with additional information about what type of process you are defining. For example, if you are defining a process to import an ASCII file, Advanced Revelation will add the prefix IMPORT.ASCII to the name you use for the process.

About ASCII file formats

ASCII file formats are generic formats that can be used by many different DOS and OS/2 applications. There are two basic variations: fixed length and variable length.

Variable-length files

Variable-length files use special characters (delimiters) to separate the fields and records in the file. Frequently used delimiters are a comma to separate fields, a carriage return-linefeed (the [Enter] key, abbreviated CRLF) to separate lines or records, and a [Ctrl-Z] (ASCII 26) to mark the end of the file.

Variable-length ASCII file

```
"Smith","Mary","123 Main Street","Aurora","CO"|}
"Thompson","John","33 Elm St","Springfield","IL"|}
"Fry","Frederick","PO Box 12","Puyallup","WA"|}
 #
```

Commas are a common field delimiter, and each line is a separate row. The characters "|}" here represent carriage return-linefeed characters (which normally do not display). The "#" character represents the non-printing end-of-file character ([Ctrl-Z]). Text data often (not always) appears in quotes.

Quote-delimited data

In variable-length files, non-numeric data is often enclosed in quotes. This is especially true if the data might contain a character that is also used to separate fields —for example, a comma:

Text characters

```
"Smith,Mary","123 Main Street","Aurora","CO"
```

The comma inside "Smith, Mary" is part of the data, not a delimiter, therefore the field is in quotes.

Text files

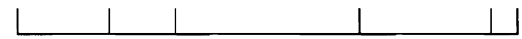
The records in a text file contain paragraphs of text instead of individual fields. This format is often used by word processing packages when you save a document as "text" or as "unformatted".

Fixed-length files

In a fixed-length ASCII file, the data is stored in cells of predetermined length:

Fixed-length ASCII file

```
Smith  Mary 123 Main StreetAurora    CO  Thomas John 33 Elm St      SpringfielIL
```



Lname (7) Fname (5) Address (15) City (10) ST (2)

A fixed-length file typically has no characters between fields or between records.

About importing ASCII files

Importing an ASCII file always begins the same way. But depending on what your ASCII file looks like, you may have to follow a different path to complete the import process. For example, a simple variable-length import makes many assumptions about your data, so you have to provide very little additional information. On the other hand, you may have to tell Advanced Revelation about special characters in your ASCII file.

The ASCII import process provides you with many options so that you can define very flexible import processes. Among the options you have are:

- Importing only selected fields, and even merging or appending those to existing data.

- Reformatting data from the ASCII file as you import it.
- Importing data in an unattended (batch) mode.
- Optimizing the import process if you have unusual import requirements.

Importing a variable-length ASCII file

Importing a simple variable-length ASCII file

The basic import process for variable-length ASCII files assumes that the ASCII file has commas between each field and a CRLF at the end of each record. An end-of-file character is optional. If you choose one of the two options for text data—Quote-delimited and Text—Advanced Revelation strips the quotes from around data as it is importing.

In a simple import, all the fields in the ASCII file are imported, and data in the Advanced Revelation table appears just at it did in the ASCII file. If you want to modify the data (for example, reformat date, time, or decimal data), or to append data onto existing Advanced Revelation columns, or import only selected fields, see "Import options for ASCII files" later in this chapter.

Before you import a variable-length file, you need to know:

- The name and location of the ASCII file.
- The name of the Advanced Revelation table into which you want to import the data. The table will be created automatically if it doesn't already exist.
- What type of data you have: comma-delimited, comma-with-quotes, or text.

To import variable-length ASCII files:

1. Choose the menu option **Utilities-Import-ASCII**.
2. At the *Import process name* prompt, enter a unique name for the process you are defining. Advanced Revelation puts the words IMPORT.ASCII in front of this name.
3. At the *DOS file name* prompt, enter the full path and DOS file name (including extensions) of the file you want to import.
4. At the *Revelation table name* prompt, enter the name of the table into which you want to import data. If the table doesn't exist, the import process will create it.
5. At the *DOS file type* prompt, press [F2] or click <Options> to see a list of common file types. Choose from one of the three pre-defined types: Comma delimited, Comma and Quote delimited, or Text.

6. When you save the process definition, you are prompted to start the import process. Choose "Yes" to start right away or "No" to store the process definition for later use.

Note To reuse an existing import definition, redisplay it in the window and press [F9] or click <Save>. That will cause Advanced Revelation to prompt you to start the import process.

Importing with non-standard delimiters

The simple import process assumes that there are commas between fields in the ASCII file, a CRLF at the end of each record, and a [Ctrl-Z] at the end of the file. It also assumes that you use double quotes around text data, if there is any.

You may encounter ASCII files that use different characters to mark fields, records, text, and the end of the file. To import data with one or more of these differences:

1. Follow steps 1 through 4 under "Importing a simple variable-length ASCII file", above.
2. At the *DOS file type* prompt, press [F2] or click <Options> and choose the option "Other". A window appears prompting for new field, record, text, and file delimiter characters.
3. Move to the appropriate prompt and enter the ASCII value of the character that you want to use. Press [F2] or click <Options> to see an ASCII chart. You may enter only one value for field, file, and text characters. You may enter two ASCII values (separated with a comma) at *Record delimiter*.

Importing a fixed-length ASCII file

Before you import a fixed-length file, you will need to know:

- The name and location of the ASCII file.
- The name of the Advanced Revelation table into which you want to import the data. If it doesn't already exist, it will be created as part of the import process.
- The starting position and length (in bytes) of each individual field in the ASCII file.
- The total length of the record (in bytes) in the ASCII file.

To import a fixed-length file:

1. Follow steps 1 and 4 for simple variable-length files, above.
2 At the *DOS file type* prompt, press [F2] or click <Options> and choose "Fixed length fields" from the popup. The Import ASCII Fields window appears.

3. At the *Field name* prompt, enter the name of the field you want to import. If the Advanced Revelation table already exists, you can press [F2] or click <Options> to show a list of defined columns. If the table doesn't already exist, the name you enter here will be used to create a column definition in the dictionary of the Advanced Revelation table.
4. If the data contains a date, currency, time, or decimal number, enter the appropriate code at the *Conversion* prompt. Press [F2] or click <Options> to see a list of available conversion codes.
5. At the *Start byte* prompt, enter the starting position of this field within the ASCII record. For example, if this is the first field in the ASCII record, enter a 1. If the field begins at the tenth character of the record, enter 10.
6. At the *Length* prompt, enter the exact length of the field.
7. Repeat steps 3 through 6 for each field in the record.
8. At the *Record length* prompt, enter the total length of the record.
9. Press [F9] or click <Save> to save the field definitions, and [F9] or <Save> again at the main window to save the import process definition and to start the import process.

Import options for ASCII files

Importing selected fields and assigning them to columns

You have the option of importing only selected fields from the ASCII file. Choosing specific fields to import also allows you to:

- Create new or temporary column definitions in the Advanced Revelation table.
- Assign fields from the ASCII file to specific columns in the Advanced Revelation table.
- Merge or append data from ASCII records to existing data in Advanced Revelation rows.
- Reformat data. This is important if you want to import date, time, decimal, or currency data.

Before you select particular fields to import, you should:

- Know the exact number of fields in your ASCII file and *which ones in particular* (by position in the ASCII record) you want to import.

- For existing Advanced Revelation tables, have all the columns already defined, or at least those into which you want to import data.
- For new Advanced Revelation tables, know the formats of the data you are importing. For example, you should know which fields in your ASCII file contain dates, which ones contain decimal numbers, etc.

Creating new column definitions while importing

You can import ASCII data into an Advanced Revelation table that doesn't already exist when you begin. In that case, the field names that you assign when defining the import process will be created as column definitions in the dictionary of the new Advanced Revelation table.

If you create new Advanced Revelation columns during the import process, the columns are assigned sequential position numbers. For example, if you define the import fields NAME, CITY, and PHONE for a new Advanced Revelation table, they will correspond to row positions 1, 2, and 3, respectively. New column definitions are stored in the dictionary of the Advanced Revelation table into which you are importing.

Reordering and skipping ASCII fields during the import process

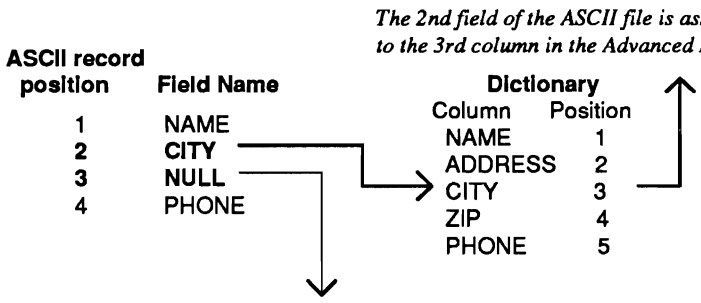
Unless you specify otherwise, the ASCII import process automatically assigns each field from the ASCII record to a corresponding position in a Advanced Revelation row. If you want to assign fields from the ASCII record to an Advanced Revelation row in a different order than that in the ASCII record, or if you want to skip over some ASCII fields while importing, you must that when you define the import process.

To assign ASCII fields to Advanced Revelation row positions, you assign an Advanced Revelation column name to each position in the ASCII record that you want to import. The import process uses the column definition in the Advanced Revelation dictionary to determine where the ASCII data should go.

Skipping fields in a variable-length ASCII file

When specifying fields to import from a variable-length file, you must provide information about each field in the ASCII file, even if you are skipping some. For example, you may only be interested in importing fields 1, 2, and 4 of your ASCII file. Nonetheless, you must provide information for the missing field (3) so that Advanced Revelation will know to skip over that field when importing. The special keyword NULL is used to indicate that you want to skip a field in the ASCII file.

Reordering and skipping fields in variable-length ASCII files



NULL means field 3 of the ASCII file will be skipped.

To import only particular fields in a variable-length ASCII file::

1. Follow steps 1 through 5 under "Importing a simple variable-length ASCII file" earlier in this chapter. Make sure you go past the *DOS file type* prompt.
2. Before saving your definition, press [Shift-F1]. This displays the Import ASCII Fields window.
3. At the *Field name* prompt, enter the column name that corresponds to the first field of the ASCII record. If you want to skip this field in the ASCII record, enter the word NULL.
4. If this is a new column definition, and if the data in the corresponding ASCII field is a date, a time, currency, or a decimal number, enter conversion information at the *Conversion* prompt. Press [F2] or click <Options> to see a list of valid conversion choices.
5. Repeat steps 3 and 4 for each field you want to import.
6. Press [F9] or click <Save> to save your definition and begin the import process. If the Advanced Revelation table already exists, you will be asked if you want to clear the existing data.

Caution! If you are merging the ASCII data with existing rows in an Advanced Revelation table, be sure *not* to clear the existing table!

Skipping fields in a fixed-length ASCII file

Follow the steps for importing a fixed-length file, and simply leave out the portions of the record that you want to skip. For example, if your ASCII records contain a company name starting at character 10 that is 25 characters long, but you don't want to import this data, leave out that range when defining the fields to import. However, you should be sure that the total record length is still correct for the record as a whole:

Column name	Conversion	Start byte	Length
Name		1	20
City		31	10
DOB	D	41	8
		Record length	48

This skips the 10 characters at 21 - 30 in the ASCII record, but the record length still reflects the skipped characters.

Merging with an existing table

If you import ASCII records that have the same keys as rows in your Advanced Revelation table, they will overwrite the existing rows. However, the merge option allows you to import ASCII records and merge them a field at a time with existing data in the Advanced Revelation table. This is useful, for example, if you have an ASCII file that has the latest version of just one column in an Advanced Revelation table.

You can only merge an ASCII file if you import selected fields. When the import process runs, it reads the selected fields from the ASCII file and updates the corresponding columns in the Advanced Revelation table.

A variation on the merge option allows you to merge ASCII data with the Advanced Revelation row only if the corresponding column in Advanced Revelation is empty.

Note See "Optimizing the import process" below for suggestions about turning off pre-sizing when importing into an existing table.

To merge:

1. Follow the steps for setting up an import process that imports selected fields.

   If you want to merge ASCII data with an Advanced Revelation column only if the column is empty, add a tilde (~) character on the end of the field name.

2. Before saving your definition, press [Shift-F3] to display the Options window.
3. At the *Merge with existing data* prompt, enter "Y".
4. Save the Options with [F9] or by clicking <Save>, then save your import process definition with [F9] or <Save> again.

Appending ASCII data onto columns

In addition to merging ASCII data with Advanced Revelation rows (see above), you can append the ASCII data onto the data that already exists in Advanced Revelation

columns. This is useful, for example, if a column in an Advanced Revelation table stores a history of client contacts and you are adding to that from an outside source.

If the column to be appended is multivalued (as defined in the Advanced Revelation dictionary), the ASCII data will be added as new values. Otherwise the data is added directly onto the existing data with no space.

Note See "Optimizing the import process" below for suggestions about turning off pre-sizing when importing into an existing table.

To append ASCII data to the end of existing columns:

1. Create an import definition that imports selected columns.
2. For each field that you want to append, add a colon (:) onto the end of the field name in the import process definition.
3. Run the process normally. Instead of overwriting existing data, the import process updates the designated columns.

Importing date, currency, decimal, or time fields

If you are importing ASCII fields with date, currency, decimal, or time data in them, you should convert this data to its corresponding Advanced Revelation format when you import the file. Saving data in Advanced Revelation's format saves space and makes it easier to manipulate. In addition, it make that data compatible with other Advanced Revelation processes such as windows, reports, etc.

To do this conversion, you *must* use the steps for importing specific fields, even if you intend to import all the fields from your ASCII records.

For more information about Advanced Revelation's data formats, see the chapter "Creating windows with Paint".

If you enter the names of existing column definitions for the fields to import, the data formatting information in the column definition is used to reformat the data while importing. For example, if one of the columns you specify for importing has been defined in Advanced Revelation as having a date format, then the import process will convert it to Advanced Revelation's internal date format when importing.

If you create new column definitions as part of the import definition, you should be sure to include the proper conversion information when listing the fields to import.

Importing data from a file with a header

A file header is a block of information at the beginning of the ASCII file that isn't part of the data to be imported. For example, a header block might contain information about what fields there are in the ASCII file, how long each one is, etc. Because the header isn't part of the data, you want to skip over it during the import process.

To import an ASCII file with a header:

1. Follow the steps for setting up the import process for the type of ASCII data you have (variable-length comma-delimited, etc.).
2. Before saving your definition and running the import process, press [Shift-F3] to display the Options window. At the *Header length* prompt, enter the size in bytes of the header to skip.

Note The header is only skipped at the beginning of the file. If you have periodic headers (for example, page headings in an ASCII file generated from a report), you must strip these out before importing the data (using a non-Advanced Revelation utility).

Interrupting and restarting an import process

You can interrupt a long import process at any time by pressing [Esc]. When you confirm that you want to quit the import process, Advanced Revelation stores information about your current position in the ASCII file.

By default, if you restart the same import process, Advanced Revelation will begin the import from the beginning of the ASCII file. However, you can set a flag to tell Advanced Revelation to pick up from where you left off last time.

Note You cannot restart a process if you want to import in Batch mode.

To restart a process that you interrupted:

1. Display the Import ASCII window and recall the import process definition you created earlier.
2. Press [Shift-F3] to display the Options window.
3. At the *Suppress restart* prompt, enter "Y".
4. Save the options with [F9] or <Save>, and then press [F9] or click <Save> to save your process definition and restart the import process. You will be asked to confirm that you want to restart where you left off.

Importing in unattended (batch) mode

The import process is usually an interactive process—you are asked questions all along as the process continues. However, if you want to run the import process while the computer is unattended (for example, overnight), you need to use a special batch mode that doesn't depend on user interaction.

Note You cannot use Batch mode to carry on with an import process that you interrupted earlier.

To set up an import for batch mode:

1. Set up the import process definition for the type of import you are doing.
2. Before saving your definition, press [Shift-F3] to display the Options window.
3. Set any options that you need, and for which the system would normally prompt you. For example, set the *Clear Revelation table* prompt to "Y" if you want the import process to clear the table before importing.

   Caution! Be sure you know what the default values are in the Options window, and what your custom settings are. Because the import process will not prompt you, there will be no warning before any potential data-destroying process (such as clearing the Advanced Revelation table first).
4. At the *Batch* prompt, enter 'Y'.
5. Save the options window, then save the import process definition.

To run a batch-mode import process:

- Display the Import ASCII window, recall your import process definition, and save it with [F9] or <Save>. You are prompted to begin the process right then.
- Alternatively, you can use the TCL command IMPORT to run the import process. See the TCL section of the *Reference manual* for details.

Optimizing the import process

Advanced Revelation's default options for the import process will normally run the import in an optimal manner. For example, Advanced Revelation pre-sizes the Advanced Revelation table based on the size of the ASCII file you are importing. Under certain circumstances, however, you might want to set options to make the import process go faster. If you are creating new Advanced Revelation columns during the import process, the columns are assigned sequential position numbers. For example, if you define the import fields NAME, CITY, and PHONE for a new Advanced Revelation table, they will correspond to row positions 1, 2, and 3, respectively.

Importing to an indexed table

If you are importing to an existing Advanced Revelation table, the table might have one or more indexes on it. If data you are importing is a big percentage of the *total* data in the table, you can speed up the *overall* process by setting an option in the import definition that:

- Temporarily removes indexing
- Imports the data
- Restores indexing

After the import processes is done, you must then rebuild all the indexes on the table. This last step can be time consuming, but if you are importing a lot of data the speed gained in the import process very often offsets the extra time spent in rebuilding the indexes.

To optimize importing of large quantities of data to an indexed table:

1. Define an import process as usual.
2. Before saving the definition, press [Shift-F3] to display the Options window.
3. At the *Remove and restore indexing* prompt, enter "Y".
4. Save the options with [F9] or <Save>, then save and run the import process with [F9] or <Save>.
5. After the import process has run, rebuild all the indexes in the Advanced Revelation table. Choose the menu option **DB Admin-Indexes-Update-Rebuild indexes in a specific table** and fill in the name of your Advanced Revelation table.

Pre-sizing the Advanced Revelation table

Advanced Revelation attempts to size the Advanced Revelation table to match the size of the ASCII file you are importing. However, if you are importing a small ASCII file into a large Advanced Revelation table, you do *not* want to pre-size the table. This recreates the existing Advanced Revelation file to match the (small) ASCII file size, and inefficiently squeezes the existing Advanced Revelation data into the smaller table.

To turn off pre-sizing for the import process:

1. Define an import process as normal.
2. Before saving the definition, press [Shift-F3] to display the Options window.
3. At the *Presize the Revelation table* prompt, enter "N".
4. Save the options with [F9] or <Save>, then save and run the import process with [F9] or <Save>.

Checking for embedded delimiters

By default, the import process checks for field delimiters within each ASCII field. It does this by looking for quote marks (or some other text delimiter) and then searching for a field delimiter within the quotes. If your data does not contain any text

delimiters, *and* you know that there are no field delimiters within your fields, you can speed the import process by bypassing this check. To do so:

1. Define an import process as normal.
2. Before saving the definition, press [Shift-F3] to display the Options window.
3. At the *Look for embedded field delimiters* prompt, enter "N".
4. Save the options with [F9] or <Save>, then save and run the import process with [F9] or <Save> again.

Creating temporary column definitions

Normally, column definitions you use in the import process are stored in the dictionary of the table to which you are importing. However, you can also create temporary columns (ones that will be thrown away after the import process). To do so:

1. Set up an import process as usual.
2. While still in the Import ASCII window, press [Shift-F3]. The Import ASCII Options window appears.
3. At the *Create new dictionary entries* prompt, enter "N". Save your option by pressing [F9] or clicking <Save>.
4. Proceed as for any other import process.

Troubleshooting the ASCII import process

If your import process did not run as you thought it would, check the following for suggestions.

Data is missing in fixed-length import

Check the starting position and lengths of fields in your ASCII file against the ones you have in your import process definition.

Data is offset by one or more characters

- Check for a header in the ASCII file that might have offset the data.
- Check for extra characters such as line breaks, even if you have a fixed-length file. If you have linebreaks or other non-data characters in a fixed-length file, you must account for them in the total record size.
- Check for multi-character field or record delimiters in a variable-length file. Advanced Revelation allows only one character for these delimiters. If you have

longer delimiters, you may have to adjust the file using a non-Advanced Revelation utility, or to import the data along with the extra characters, and then clean it up inside Advanced Revelation. Both options may require programming.

- If a fixed-length file is off by just one or two characters, you can adjust the starting offset for the file. In the Import ASCII window, recall your import process definition and press [Shift-F3]. At the last prompt, *Offset*, enter a value for the number of characters by which to offset the file forward or backward (use a negative sign if necessary).

Data is missing in columns

If you seem to be missing data in some of the columns, create a listing of the Advanced Revelation dictionary for the table to which you are importing (**DB Admin-Columns-List**). If more than one column definition used in your import has the same position (example: two of the columns are for position 3 in the row), you are overwriting data from one column with data from another. Change the import process definition to reference only columns that have unique positions.

Importing is slow

Review the discussion under "Optimizing the import process" above. The following options have a big influence on the speed of the import process:

- Importing selected fields.
- Checking for text and embedded delimiters.
- Importing into an indexed table.
- Not pre-sizing, or inappropriately pre-sizing the Advanced Revelation table.

Importing a dBASE III file

The dBASE import process copies data from a dBASE III file into an Advanced Revelation table. Because Advanced Revelation can read the structure of the dBASE III file, you need to provide only the most basic information.

If the Advanced Revelation table does not exist, it is created as part of the import process. The import process will also create new dictionary entries in the Advanced Revelation table for the fields in the dBASE III file. The new column definitions are appended onto any existing definitions in the dictionary. The import process then uses these new column definitions to determine where to put the data in the Advanced Revelation table.

Because each row in an Advanced Revelation table must have a unique key (the dBASE III file doesn't necessarily have to), you must know what field in the dBASE

file can serve as a key. The field you select must contain have unique data, or you will overwrite some of the rows you are importing. For example, if you select the field LASTNAME as a key, but there is more than one SMITH in the file, each SMITH will be overwritten by the subsequent ones in the file.

If you don't know what field to select as the key, you can choose to use a sequential counter, which guarantees that every row will have a unique key.

To import a dBASE III file:

1. Choose the menu option **Utilities-Import-dBASE III.**
2. At the *Process name* prompt, enter a unique name for the process you are defining. Advanced Revelation adds the prefix IMPORT.DB3.
3. At the *Dbase file* prompt, enter the path and full name (including extension) of the dBASE file you want to import. Advanced Revelation immediately fills in the structure information.
4. At the *Revelation table* prompt, enter the name of the table into which you want to import.
5. Press [F9] or click <Save> to save your process definition and start the conversion. If the Advanced Revelation table already exists, you are asked if you want to clear it before importing.
6. You are prompted to identify a field from the dBASE file to serve as primary key for imported rows. If no field in the dBASE file is guaranteed to contain unique values, select the option "Sequential counter", which will create a numeric key for each imported row.

Importing a Lotus 1-2-3 spreadsheet

You can import an entire Lotus .WKS file into an Advanced Revelation table. During the import process, rows in the spreadsheet becomes rows in the Advanced Revelation table, and columns in the spreadsheet become columns in the table.

New rows in the table are assigned a sequential number to match their row number in the spreadsheet. Columns are also imported sequentially—column A in the spreadsheet becomes the first column in the Advanced Revelation table, column B becomes the second column, etc.

If the Advanced Revelation table does not already exist, it will be created. The import process builds dictionary entries for the cells it imports, even if there already are column definitions in the dictionary.

- If the Advanced Revelation table already exists, the import process asks you whether you want to clear the dictionary and the data portions of the table.

- If you clear the dictionary, the *only* column definitions will be the ones that the import process builds.
- If you clear the data portion of the table, the only data in the table will be the imported rows.
- If you do not clear the data portion of the file, imported rows overwrite existing rows of the same key in the Advanced Revelation table, but other rows are left intact.

Note The Lotus 123 import process imports the entire spreadsheet. If you want to import only a portion of it, use Lotus 1-2-3 to copy the cell range you need to a new spreadsheet and import the second spreadsheet instead.

To import a spreadsheet:

1. Choose the menu option **Utilities-Import-Lotus 123**.
2. At the *Import template name* prompt, enter a name for the import process you are defining. Advanced Revelation adds the prefix IMPORT.123.
3. At the *Lotus 123 file name* prompt, enter the path, name, and extension (WKS) of the spreadsheet to import.
4. At the *Revelation table name* prompt, enter the name of the table into which you want to import the spreadsheet.
5. Press [F9] or click <Save> to save your definition and start the import process. If the Advanced Revelation table exists, you will see prompts that ask you whether you want to clear the dictionary and data portions of the table.

37. Exporting data out of Advanced Revelation

About exporting

Exporting data allows you to copy individual rows from an Advanced Revelation table to a file that can be used by some other process. You can export to these file formats:

- dBASE III
- Lotus 1-2-3 (WKS format)
- ASCII files (fixed- and variable-length)

Note Remember that you can use Environmental Bonds to share data with other applications directly (without importing or exporting). For more information, see the section on Environmental Bonding.

Because of the flexible table structure in Advanced Revelation, you may find that some data does not export into other environments very well. In particular, you will likely have problems if you attempt to export the following types of data:

- Columns that contain large quantities of text. Most other environments have much more restrictive limitations on the size of a column than Advanced Revelation does.
- Multivalued columns. Advanced Revelation exports multivalued columns as they appear in your data table, but other environments can rarely use them in this format.

Because Advanced Revelation uses the dictionary of the table you are exporting, you can export any column, including columns with symbolic definitions (formulas).

Creating export definitions

To export data, you use the Export window to define an export template that tells Advanced Revelation such information as what format you want to export to, what operating system file to create, and what columns from the Advanced Revelation table to export. To run the export process, you press [F9] or click <Save>. Advanced Revelation then saves your export definition, and asks you if you want to export the data right away.

You can re-use your export templates as often as necessary. This would be useful, for example, if you regularly export the same columns from the same table.

You can also run an export process from the Command window. You must run the export process from the Command window if you want to export only selected rows (see below).

About formatting exported data

When exporting data, Advanced Revelation uses the definition of the column in the Advanced Revelation dictionary to determine the correct format and justification of the exported data. For example, if you export a column that has an output format for a date in Advanced Revelation, the export process will put the data into the exported file using the date format that you've defined for that column. If you have defined the data to be right-justified, the data will appear in the output file with that justification.

As a consequence, you should be certain that the output formats of the columns you want to export accurately reflect the type of data is in the column, and how you want it formatted in the export file.

Exporting selected rows

If you want to export only selected rows, you will define the export process appropriate to the type of file you are exporting to. When you save the export definition, Advanced Revelation asks whether you want to run the export process immediately. Answer "N".

Display the Command window by pressing [F5]. Use the TCL SELECT command (or GETLIST or another process) to create a select list of the rows you want to export). When the select list is active, enter this command at the Command window:

```
EXPORT type export_name
```

where *type* is the type of file to which you are exporting, and *export_name* is the name of the export definition you created earlier. Possible values for *type* are:

`ASCII`	Export to a fixed-length or variable-length ASCII file.
`DB3`	Export to a dBASE III file
`123`	Export to a Lotus 1-2-3 spreadsheet.

Creating and running an export process

Exporting to a dBASE III file

Advanced Revelation allows you to export all or selected columns from an Advanced Revelation to a dBASE III DBF file.

Note You can use the dBASE Environmental Bond to copy rows directly from Advanced Revelation tables to dBASE III files without exporting. See the Environmental Bonding section for details.

You can export multivalued columns to a dBASE file, but these will not be handled properly in the dBASE environment.

Before you export data from an Advanced Revelation table to a dBASE III file, you will need to know:

- The name of the Advanced Revelation table that you want to export.
- The name of the dBASE III file to which you are exporting. The file does not need to exist before you start.
- The names of the columns in the Advanced Revelation table that you want to export. You are not required to export all the data in the table.
- The data type of each column you want to export. You will be required to identify each column as one of these dBASE data types:

  Character
  Date
  Logical ("Y" or "N")
  Memo
  Numeric (no punctuation)

- The length of the fields that you will be creating in the dBASE file. Advanced Revelation will truncate data to fit into the field lengths that you specify.
- For numeric fields, the decimal count (precision) of the values you are exporting.

To run the export process:

1. Choose the menu option **Utilities-Export-dBASE III.**
2. At the *Process name* prompt, enter a unique name for the process you are defining. Advanced Revelation adds the prefix "EXPORT.DB3."
3. At the *Dbase file name* prompt, enter the name of the file to which you are exporting. Include the extension .DBF. If the file does not already exist, it will be created automatically during the export process.
4. At the *Revelation table* prompt, enter the name of the table you are exporting.
5. For each column, at the *Column name* prompt enter the name of the column to export.
6. At the *Field type* prompt, enter the first character of the dBASE data type that you are assigning to this column.
7. At the *Field length* prompt, enter the size that the column should be in the dBASE file. Data longer than this size will be truncated during the export process.
8. If the column contains all numeric data, enter at the *Decimal count* prompt the number of decimal places you want to assign to the data.

9. Press [F9] or click <Save> to save your process definition and start the export process. If the dBASE file to which you are exporting already exists, you must indicate whether you want to overwrite the existing file.

Exporting to a Lotus 1-2-3 spreadsheet

Please read "About exporting" earlier in this chapter for information on creating an export definition and on exporting selected rows to a 1-2-3 spreadsheet.

When you export to a Lotus spreadsheet, Advanced Revelation turns rows of Advanced Revelation data into rows in the spreadsheet, and data columns into spreadsheet columns. The result is a spreadsheet in 1-2-3 that begins in cell A1. (If you want to put the Advanced Revelation data into a particular cell range in 1-2-3, you should then copy the exported spreadsheet into a new spreadsheet using the 1-2-3 cell copy utility.

The length of the cells in the resulting spreadsheet is based on the display length of the columns you export, as stored in the Advanced Revelation dictionary.

Before you export to a Lotus 1-2-3 spreadsheet you should know:

- The name of the Advanced Revelation table you want to export, and the names of the columns you want to export.
- The name of the file that you want to create during the export process. The export process will create the file if necessary. If the file already exists, it will be overwritten.

To export to a spreadsheet:

1. From the Development menu, choose **Utilities-Export-Lotus 123**. The Export Lotus 123 window appears.
2. At the *Export template name* prompt, enter a name that you want to assign to this export definition. The window adds the prefix "EXPORT.123." to your name.
3. At the *Lotus 123 file name* prompt, enter the name of the file that you want to create during the export process.
4. At the *Revelation table name* prompt, enter the name of the Advanced Revelation table to export. Press [F2] or click <Options> for a list of available tables.
5. At the *Column name* prompt, enter the names of the columns you want to export, one per line. If you press [F2] or click <Options>, a multi-select popup of all the columns in the Advanced Revelation table displays.
6. Press [F9] or click <Save> to save the export definition. Advanced Revelation asks whether you want to run the export process immediately. If you want to export the entire Advanced Revelation table immediately, answer "Y". If you want to run the export process later, or if you want to export only selected rows, answer "N".

Exporting to a variable-length ASCII file

For information about ASCII file formats, please see the chapter "Importing data into Advanced Revelation". For information about export definitions, please see "About exporting" earlier in this chapter.

Before exporting to a variable-length ASCII file, you will need to know:

- The name of the Advanced Revelation table to export, and of the columns to be exported from that table.
- What characters you want to use in the ASCII file to separate fields and records from one another.
- What character (if any) you want to use to mark the end of the ASCII file.

For information about exporting only selected rows to an ASCII file, see the topic "Exporting selected rows" earlier in this chapter.

To export to a variable-length ASCII file:

1. From the Development menu, choose **Utilities-Export-ASCII**. The Export ASCII window appears.
2. At the *Process* name prompt, enter a name for the process definition you are creating. Advanced Revelation adds the prefix "EXPORT.ASCII." to your name.
3. At the *DOS file name* prompt, enter the name of the file to which you are exporting. If the file doesn't exist, Advanced Revelation will create it. If the file already exists, it will be overwritten during the export process.
4. At the *Revelation table* name prompt, enter the name of the table that you are exporting. Press [F2] or click <Options> for a list of available tables.
5. At the *Field delimiter* prompt, enter the ASCII value of the character that you want to use to separate the fields in the ASCII file. For example, to indicate a comma, enter the value "44". You can enter only one value here. To see an ASCII chart from which you can choose the value, press [F2] or click <Options>.
6. At the *Record delimiter* prompt, enter the ASCII value of the character or characters that you want to use to separate records in the ASCII file (press [F2] or click <Options> to see an ASCII chart). You can enter one or two values here. If you enter two values, separate them with a comma. For example, to indicate a delimiter of carriage return-linefeed, enter:

```
13,10
```

7. At the *End-of-file* delimiter prompt, enter a single ASCII value to indicate what character should be put at the end of the ASCII file. The default is ASCII 26, the [Ctrl-Z] character.
8. Skip the *Record length* prompt, isn't used here.

9. At the *Column name* prompt, enter the names of the columns that you want to export, one per line. Press [F2] or click <Options> to see a multi-select popup of all the columns in the Advanced Revelation table. Skip the *Start byte* and *Length* prompts for each column—these are only used for fixed-length ASCII files.
10. Press [F9] or click <Save> to save your export definition. Advanced Revelation asks whether you want to run the export process immediately. If you want to export the entire Advanced Revelation table immediately, answer "Y". If you want to run the export process later, or if you want to export only selected rows, answer "N".

Exporting to a fixed-length ASCII file

For information about ASCII file formats, please see the chapter "Importing data into Advanced Revelation". For information about export definitions, please see "About exporting" earlier in this chapter.

Before exporting to a fixed-length ASCII file, you will need to know:

- The name of the Advanced Revelation table to export, and of the columns to be exported from that table.
- The length of each field that you will be creating in the ASCII file. For example, if you are exporting a column called COMPANY_NAME, you need to know exactly how many characters you want the company name to occupy in the ASCII file.

If you want to export only selected rows, please see the topic "Exporting selected rows" earlier in the chapter.

To export to a fixed-length ASCII file:

1. From the Development menu, choose **Utilities-Export-ASCII**. The Export ASCII window appears.
2. At the *Process* name prompt, enter a name for the process definition you are creating. Advanced Revelation adds the prefix "EXPORT.ASCII." to your name.
3. At the *DOS file name* prompt, enter the name of the file to which you are exporting. If the file doesn't exist, Advanced Revelation will create it. If the file already exists, it will be overwritten.
4. At the *Revelation table* name prompt, enter the name of the table that you are exporting. Press [F2] or click <Options> for a list of available tables.
5. Skip the *Field delimiter*, *Record delimiter*, and *End-of-file delimiter* prompts, because these apply only to variable-length ASCII files.
6. Skip the *Record length* prompt—this isn't currently used in the export process.

7. At the *Column name* prompt, enter the names of the columns you want to export. Press [F2] or click <Options> to display a multi-select popup of all the columns in the Advanced Revelation table. Enter the columns here in the order that you want them to appear in the ASCII file.
8. For each column you enter here:
   - Skip the *Start byte* prompt, which isn't currently used.
   - At the *Length* prompt, enter the number of characters that you want this column to occupy in the ASCII file.
9. Press [F9] or click <Save> to save your export definition. Advanced Revelation asks whether you want to run the export process immediately. If you want to export the entire Advanced Revelation table immediately, answer "Y". If you want to run the export process later, or if you want to export only selected rows, answer "N".

38. Building macros

This chapter explains how to define and use customized sets of macro keys to simplify and speed up operations and procedures in Advanced Revelation.

Defining and using macro sets

The keys [Alt-1] through [Alt-5] have been reserved for use as macro keys. Each of these keys can be programmed so that it performs a special operation any time it is pressed. Any TCL or R/BASIC command can be assigned to a macro key, placing often used commands at your fingertips.

When to define macro keys

Macro keys provide operating shortcuts. If you frequently type a particular command at the Command window, consider defining a macro key to issue the command for you. Define and use macro keys to streamline the operation of activities that you do often.

If a more complicated procedure involving many operations will be performed routinely, use the options on the Capture menu to record and duplicate it.

Creating a macro set

The operations you attach to the five macro keys are stored in a macro set. To create a macro set, press [Alt-M] to display the Macros window, or choose the option **Utilities-Developer-Macro** from the Development menu.

Give the macro set an identifying name. Assign a code and command to each key in the set to define the operations that the keys will perform. Press [F9] or click <Save> to save the information contained in the macro set.

Each macro set that you define and save becomes an entry in the MACROS table. You can define as many macro sets as you like, although only one macro set can be active at a time.

Environment

Use the General System Configuration window to activate macro building, to activate macro execution, and to determine which macro set is active when you log on to Advanced Revelation. To display the General System Configuration window, choose the **Options-Configuration-Environment-General** option from the Opening menu.

Accessing the Macros window

To access the Macros window use either of these methods:

- Press [Alt-M].
- Choose the **Utilities-Developer-Macro** option from the Development menu.

When the Macros window appears, it displays the active macro set, identifying it by name. At this point you can do a number of things:

- Use the Macros window to create a new macro set.
- Add to or edit the key functions in the active macro set.
- Display another macro set, making it the active set.

Defining a macro key

For more information, see "Using codes and commands" in the *User's Guide*.

Each macro key is defined by using a **code and command**. Below are some of the codes used frequently for macro keys.

When you want to	Choose code
Run a TCL command that will use or create a row key list	X
Run a TCL command that will not use or create a row key list	E
Reproduce the keystrokes used to carry out an operation	K
Display a window	W
Display a menu	M
Run an R/BASIC subroutine	S

Using the keystroke translator to define a macro key

You can program a macro key to reproduce a short series of keystrokes that control some operation. Use code K (Macro Keystrokes). After entering K at the Code prompt,

tell Advanced Revelation the names of the keystrokes in the macro by using the keystroke translator.

The keystroke translator

[Ctrl - –] ([Ctrl + hyphen]) is a toggle switch that turns the keystroke translator on and off. Type [Ctrl - –] to turn on the translator. There will be a delay while the translator is loaded into memory. Wait a few moments, then type the sequence of keystrokes that you want to reproduce whenever the macro key is pressed. When the translator is on, it processes each keystroke and writes out the name of each key in a format that Advanced Revelation recognizes. The keystroke translator remains active until it is toggled off by pressing [Ctrl - –] again.

For more information on recording, viewing, and playing back a keystroke sequence, see "Capturing Keystrokes" in the *User's Guide*

Alternatively, use the Capture utility to record and view the keystrokes that should be entered at the Macro Keystroke Command prompt. When a longer keystroke sequence will be performed routinely, use the Capture utility to record and play it back.

Making a macro set active

To change the active set through the Macros window, press [Al;t-M] to display the window, then type in the name of the macro set that you want to activate and press [Enter]. As soon as the row is displayed, it becomes the active macro set.

Another way to change the active set is to enter the MACRO command at the Command window. Use the following syntax:

```
MACRO macro.name
```

As soon as the command is entered, the named set becomes active.

39. Capturing keystrokes

When to use keystroke capture

The Capture utility enables you to record and play back long keystroke sequences. If you perform an activity routinely, you can save execution time and simplify operation by recording the procedure in a capture set. Once captured, the keystroke sequence in the capture set can be played back on demand. Capturing keystrokes is useful if:

For more information on Macro Keystrokes, see "Building macros" in the *User's Guide.*

- A complicated procedure will be performed routinely.
- You want to create self-running demos and tutorials for Advanced Revelation applications. An entire procedure can be run, stopped at key points, and then continued.
- If you want to produce documentation for an application, ensuring that all steps in the process are covered. Since the Capture utility records the keystrokes used to perform an operation, it can serve as a reference later.

A capture set can be simple or complex. You can create individual capture sets or you can link capture sets by using the Editor, or by specifying code and command options. When capture sets are linked to one another, they can produce a more complex result like a self-running demo program or a tutorial. You must use more advanced techniques to link and modify capture sets. This may include writing a program to link a series of keystroke sets, or using the Editor to merge one capture set into another.

Note It is important to remember that playing a sequence of keystrokes produces useful results only if it is started under the same conditions and at the same point in Advanced Revelation where the initial capture began. This can be controlled by starting keystroke capture and playback by pressing [F5] (Command window) and then [F10] (Menu).

Capturing keystrokes

To run the Capture utility, choose the **Utilities-Developer-Capture** option from the Development menu.

You will find that it is more practical to use the special macro keys to run most of the Capture options. This avoids capturing the keystrokes that must be used to exit and access the Development menu and to select the End Capture option. It also allows you to control more accurately the point at which keystroke playback begins.

Keystroke capture operations

When you want to	Press	Or choose menu option
Start capturing keystrokes	[Alt-0]	Start
End capturing keystrokes	[Alt-0]	End
Suspend a capture sequence	[Alt-8]	
Play a captured set of keys	[Alt-9]	Play
Modify a captured set of keys		Modify
Stop capturing keys without keeping them		Cancel

Note Part of your captured keystroke sequence may involve making selections from menus. When capturing selections off a menu, always type the first character of the menu option to select the item. This ensures that when the capture set is played back, the proper menu selections will be made.

Never move the cursor to make a menu selection, as the resulting capture set may play back differently each time it is run. If you do, the menu defaults that are active during keystroke playback can divert the keystroke sequence to a different path, resulting in meaningless processing. This is because the active default menu options at the time of playback vary, based on earlier menu selections made during the Advanced Revelation session. The starting point (default menu option) from which the cursor will move in a menu is not predictable.

Suspending the capture process

Use [Alt-8] to temporarily suspend the capture process. The current keystroke set will display in the full screen Editor, allowing you to add to and edit the set.

Editing captured keystroke sets

You can edit the captured keystroke sequence using the editor. This permits you to remove or change keystrokes that you might have made in error. Invoke the editor (**Define-Program-Edit** from the Development menu, or [F5] to display the Command window, then EDIT). Use the table name of CAPTURED, and at the *Row(s)* prompt, enter the name of the captured keystroke sequence you want to edit. Press [F2] or click <Options> to see a list of all captured keystroke entries.

Adding text

You can also add text into the captured keystroke set. By adding text, you can make use of features of the captured keystroke process that you could not otherwise access.

Adding comments

You can add comments to the captured keystroke set. This enables you to insert remarks or notes into your captured keystroke sequence for later reference. An asterisk at the front of a line in the keystroke sequence is used to identify a comment. When an asterisk is entered, all text that follows it to the end of the line is treated as a comment, and will be ignored by the captured keystroke replay process.

Adding processing

You can also add code and command sets into the captured keystroke set. The code and command should be on its own separate line, and should have the format @code command@.

While in the Editor, you can use the keystroke translator to add keystrokes to a capture set. Press [Ctrl-hyphen] to toggle the keystroke translator on and off.

Playing back a capture set

A capture set can be replayed as often as needed. Be sure to start replay under the same conditions and at the same point in Advanced Revelation where the initial capture began.

To play back a set of keystrokes from any point, press [Alt-9]. Alternatively, if the context is appropriate, choose **Play** from the Capture menu. The Play Capture window appears.

You are asked to name the capture set to play. You can press [F2] or click <Options> to see a list of existing capture sets. Whatever keystrokes were captured in the named set will be replayed immediately.

Disabling keystroke capture

The ability to capture sets of keystrokes depends on information maintained in the General System Configuration window. This allows you to disable keystroke capture for any given application or user.

To display the window, choose **Options-Configuration-Environment-General** option from the Opening menu. Then press [Shift-F1] to choose the application or user for whom you want to disable keystroke capture.

Enter "N" at the *Enable keystroke capture* prompt to disable keystroke capture. If Capture is not activated, you will not be able to start capturing keystrokes but you will be able to play back existing captured sets.

Changing playback speed

You can set the keystroke playback delay so that keystroke sequences play back faster or slower for any given applications or user. From the Opening menu, choose **Options-Configuration-Environment-General** to display the General Environment window. Press [Shift-F1] to select the application or user for whom you want to reset playback delay. To reset the delay, change the value at the *Keystroke playback delay* prompt.

Index

Index

$

@

A

B

C

D

E

F

G

H

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

RevelationTechnologies

181 Harbor Drive
Stamford CT 06902
(203) 973-1000
(800) 262-4747

Revelation Technologies (UK), Ltd.
270 Upper Fourth Street
Central Milton Keynes
MK9 1DP England
0908-233-255

300-2000-00