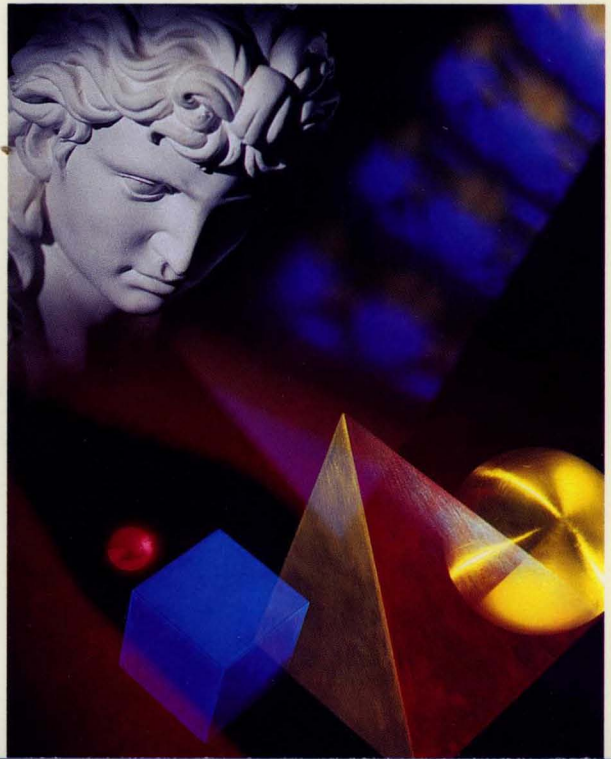
OBJECTVISION™
FOR WINDOWS

REFERENCE



BORLAND

ObjectVision™ Reference

BORLAND INTERNATIONAL, INC. 1800 GREEN HILLS ROAD
P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001

PRINTED IN THE U.S.A.
10 9 8 7 6 5 4 3 2

R2

C O N T E N T S

T A B L E S

F I G U R E S

PART 1

ObjectVision features

INTRODUCTION

This book is the second of two ObjectVision manuals:

- *ObjectVision Tutorial* tells you how to get ObjectVision up and running, and includes step-by-step examples you can use to recreate *Order*, one of the sample applications included with ObjectVision.
- This book, *ObjectVision Reference*, explains how to design ObjectVision applications and how your applications are used. It goes into thorough detail about each area of ObjectVision.

We recommend that you read the *ObjectVision Tutorial* before you begin this book.

What's in this manual

This manual is divided into two parts:

Part 1: ObjectVision features

- **Chapter 1, "ObjectVision: An overview,"** describes basic ObjectVision concepts and the benefits of creating ObjectVision applications.
- **Chapter 2, "Getting started,"** explains how to install and start ObjectVision, and briefly introduces Windows.
- **Chapter 3, "Using an ObjectVision application,"** tells you how users fill in your application's forms and how you can design easy-to-use forms.
- **Chapter 4, "Designing an application,"** describes a general process you can follow when creating ObjectVision applications.
- **Chapter 5, "Using the Form Tool,"** explains how to create, modify, and assign properties to your application's forms with the Form tool.

- **Chapter 6, "Using the Tree Tool,"** tells you how to add decision-making logic to your form's fields so that ObjectVision can calculate values for those fields.
- **Chapter 7, "Writing expressions,"** covers creating and modifying your logical expressions in both the Condition dialog box and the Conclusion dialog box.
- **Chapter 8, "@Function commands,"** describes using ObjectVision @functions to calculate a field's value. It also lists all available @functions.
- **Chapter 9, "Using the Stack Tool,"** explains how you can structure and revise the order of your application's forms using the Stack Tool.
- **Chapter 10, "Linking to data files,"** tells you how you can transfer information between your ObjectVision application and files written by other applications, such as Paradox, dBASE-compatible databases, Windows applications, and applications that produce ASCII files.
- **Chapter 11, "Distributing your applications,"** describes how you can distribute your ObjectVision applications to users.

Part 2: Appendixes and Glossary

- **Appendix A, "Keyboard and mouse operations,"** lists keys you can press and mouse techniques you can use for ObjectVision operations.
- **Appendix B, "Application limits,"** specifies ObjectVision technical limits and precisions, such as the maximum number of system function arguments. It also includes an explanation for tracking the size of your application as you design it.
- **Appendix C, "The ANSI character set,"** lists the complete set of ANSI characters and their corresponding codes.
- **Appendix D, "Configuring the Paradox Engine,"** describes how to set up ObjectVision to use the Paradox Engine on your network.
- **Glossary** gives definitions of key words used in this manual and in ObjectVision.

Typography and naming conventions

These manuals use special typefaces to help you distinguish between text you type, glossary terms, keys you press, and file names.

`Monospaced type`	This typeface represents text as it appears onscreen, as well as anything you must type.
Italics	Italics are used to emphasize certain words and to introduce new terms, which are defined in the Glossary. We also use this typeface to indicate argument names, and ObjectVision application names; for example, "The *Order* sample application."
Keycap	This special typeface represents a key on your keyboard. It often indicates a particular key you should press; for example, "Press *Del* to erase the character to the right of the pointer."
ALL CAPS	All caps are used to represent DOS directories and file names, link names, database file names, and reserved words in functions.
SMALL CAPS	@Function names are printed in small capital letters.

Late-breaking news

If there have been any changes or additions to ObjectVision since this manual was printed, you can read about them in the README.TXT file provided on your master disk.

To read the file, use a DOS or Windows word processor to open the file on the original distribution disk or in the ObjectVision directory. README.TXT is copied to your ObjectVision directory during installation.

You can print the README.TXT file for easy reference.

How to contact Borland

You must send in your Borland product registration to become a registered owner and receive technical support.

The best way to contact Borland is to log on to Borland's Forum on CompuServe: Type `GO BORAPP` from the CompuServe main menu. To locate the ObjectVision Forum follow the onscreen instructions. Leave your questions or comments there for the

support staff to process. If you need assistance with CompuServe, type the command `help`.

If you prefer, you can write a letter with your comments and send it to

Borland International
Technical Support Department
1800 Green Hills Road
P.O. Box 660001
Scotts Valley, CA 95067-0001, USA

(408) 438-5300

You can also telephone our Technical Support department between 6:00 a.m. and 4:45 p.m. Pacific Standard Time.

Whichever method of contact you choose, you must provide the following information:

- Product name and serial number on your original distribution disk. Please have your serial number ready, or we won't be able to process your call.
- Product version number. The version number for ObjectVision is displayed when you first load the program, before you press any keys.
- Computer brand and model, and the brands and model numbers of any additional hardware.
- Contents of your AUTOEXEC.BAT and CONFIG.SYS files.
- The specific steps that reproduce your problem.
- A daytime phone number where we can contact you.

C H A P T E R

1

ObjectVision: An overview

This chapter, which explains how users work with ObjectVision applications, is intended to help you understand ObjectVision design issues. It also presents an overview of ObjectVision concepts, which are explained in more detail in later chapters.

ObjectVision concepts

Applications you create with ObjectVision present a familiar forms interface to users. Your applications can include standardized procedures and decision processes users need to follow when filling in the required data. You can design applications to gather and present information for a great variety of user needs.

Your applications can contain either a single form or multiple forms. You can determine which portions of your applications have automatic completion and in what sequence users complete your applications. You can also link your applications to data files from other applications and use that data in your applications' decision processes.

Your applications can automate any task that requires users to follow complex procedures and accurately capture data, thus rendering the task relatively effortless. Some typical areas for ObjectVision applications are

- Loan/credit approval
- Insurance risk/rate determination

- Insurance claims processing
- Personnel benefit administration
- Customer support
- Order configuration/quote generation
- Telemarketing support
- Equipment diagnosis and repair
- Factory quality/yield control
- Factory scheduling decisions
- Process startup/shutdown procedures
- Purchasing decision support
- Tax reporting/planning decisions
- Work-flow coordination/routing decisions

Users

ObjectVision is easy to use for both designing and using applications.

- ObjectVision can be used to create an application for the specific requirements of an individual or a group. Programming skills are *not* necessary to create a ObjectVision application. Application designers can be experienced personal computer users or systems analysts familiar with the application task. Application users can easily modify applications for specific requirements.
- ObjectVision Runtime lets end users complete ObjectVision applications without letting them modify an application.

People who are familiar with an application's requirements can be involved in its design and maintenance. Application designers can easily incorporate user suggestions resulting in custom applications that meet the needs of the particular organization.

Forms

ObjectVision uses a forms interface because it is a familiar way to collect and display information. You can specify how your application's information will display within the forms' fields.

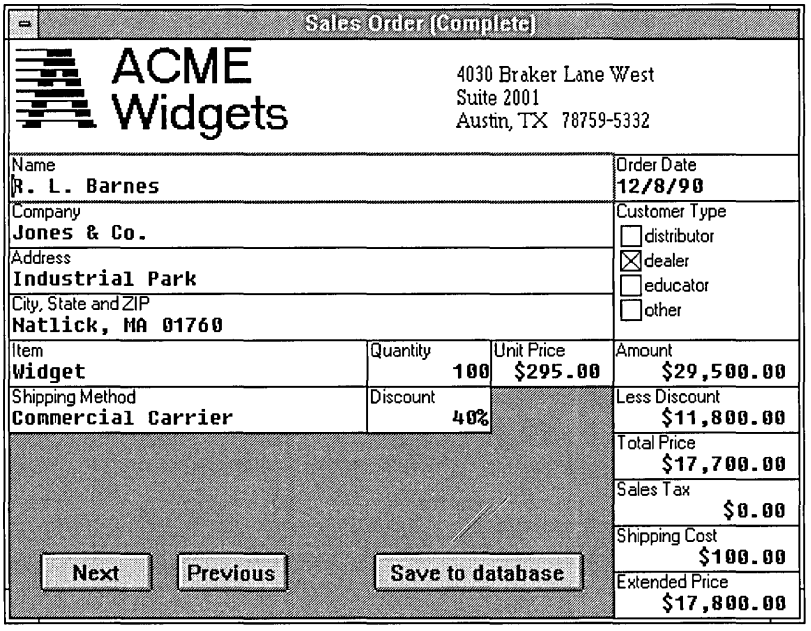
If you are familiar with spreadsheet applications, you can think of a form as a portion of a spreadsheet and the collection of forms as

the entire spreadsheet. However, ObjectVision has formatting options that are unavailable in spreadsheets, as shown in Figure 1.1.

Your blank form contains fields the user both fills in and uses as a guide for making decisions about a specific topic area. After users fill in all the fields, the completed form documents the gathered data and makes it easy to share the information with others.

Since most organizations already use paper forms, your applications can use existing forms as a model. Your ObjectVision forms can be exact replicas of or enhancements to existing paper forms. For applications that lack paper forms, you can quickly design ObjectVision forms to contain the relevant information.

Figure 1.1
The Sales Order form in the Order sample application



Tools

As you create an ObjectVision application, you use the ObjectVision Form Tool to define your forms. You can specify a variety of field properties including numeric formats, fonts, filled rectangles, help information, and borders. You use the Stack Tool to arrange the forms for easy data entry, and you use the Link Tool to connect your application to external data files.

Distribution

After you define and save a form set for your application, you can make copies of your disk file and distribute them to users. Users can then run your application using ObjectVision or ObjectVision Runtime.

For more information about licensing ObjectVision Runtime, see the *ObjectVision Runtime* brochure included with your ObjectVision materials.

Using forms

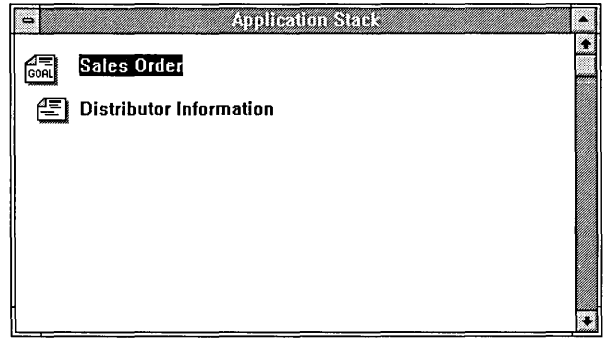
ObjectVision applications automatically provide *guided completion*, which makes it easy for users to fill in the data required by the application. Guided completion selects only the fields requiring users to supply information. The user can interrupt guided completion at any time to open, close, or rearrange forms on the display.

A stack is equivalent to a stack of paper forms.

The set of forms in an application is ordered in a *stack*. Figure 1.2 shows the stack of forms for the *Order* sample application. Each form is represented as an icon, and its unique title appears to the right of the icon.

The form at the top of the form stack is the Goal form, which is the initial form that appears when the application opens. If a form is manually selected by a user, the initial field sequence is interrupted and new guided completion is determined for the selected form. or the field sequence, that form moves to the front of the screen and to the top of the stack. The user can only enter information into the active form, even when other forms are visible under it.

Figure 1.2
Order's stack of forms



Fields

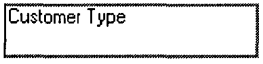
Fields are the blanks on your forms that contain either user-entered or ObjectVision-calculated values. You supply a unique name for each ObjectVision field in your application. If you are familiar with spreadsheets, you can think of a field as a named spreadsheet cell.

You can put the same field on multiple forms and in multiple locations on a form. You might want to place certain fields (for example, a client's name or identification number) on a number of forms to save users the trouble of entering the same information repeatedly. When a field is put on multiple forms, it has the same value on all forms, regardless of which form the user actually enters the information into.

Highlighting different fields

ObjectVision distinguishes among different types of fields by displaying different highlights around the field border:

- A dashed line within a heavy line appears when a user selects one of your form's calculated fields.



- A heavy line appears when either ObjectVision or a user selects a field that requires user input.
- A field you have protected from user input displays the heavy line highlight without a text pointer.

Overriding values

Users can *override* unprotected, calculated fields in your application's forms by entering a new value. When a user overrides the calculated value, the value the user enters replaces the calculated value, without removing the decision tree logic you assigned to that field.

After a user overrides a field, a dot pattern appears in that field along with the value to help remind the user they have changed your calculated field value. Users can use the Field | Calculate command to recalculate the value based on the decision tree logic.

Values

4,096 is the maximum number of characters allowed in a field value.

Field *values* are undefined until the user enters a value, ObjectVision calculates a value, or a link to external data provides a value. Any field can contain any type of value, such as a numeric, text, or error value.

A field can have only one value at a time. When the user enters a new field value, the old value is replaced.

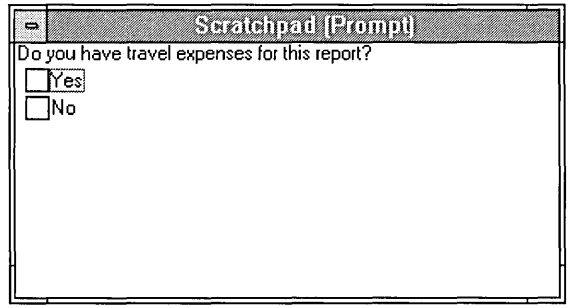
Scratchpad forms

Scratchpad forms are temporary forms created automatically by ObjectVision.

You don't have to include all your fields on a form. ObjectVision automatically displays a *Scratchpad form* to prompt users for fields that are not put on any form.

For example, suppose you are modeling your application on existing forms that lack some or all of the background information needed to calculate a value. Rather than add a field to one of your forms or create a new form, let ObjectVision prompt the user for the field's value using a Scratchpad form.

Figure 1.3
Scratchpad form in the Expense sample application



Guided completion

When users select a field, they interrupt guided completion.

By default, ObjectVision determines the field selection sequence from your application's form design. The field selection sequence guides users only to fields requiring their input. Users need to fill in only those fields that are selected as part of guided completion.

Guided completion helps users enter values required to complete the Goal form by proceeding from the top left corner of your Goal form to the bottom right corner of your Goal form.

The Goal form is the top form in the stack when users open your application, but other forms requiring user input can temporarily appear during form completion.

When a calculated field on your Goal form lacks a value, ObjectVision determines which other field values are needed to calculate that value. If the next field in the sequence is on a different form, the new form appears temporarily to receive user input.

If a field gets its value from either a decision tree or an external link, it is not selected during guided completion.

Form status, such as (Goal), displays in parentheses after the application name.

Users can interrupt guided completion to select any form in your application. Any user-selected form that is incomplete becomes the *Goal form*. This form status indication appears in the application title bar after the application name.

Form status (Complete) displays after all required fields are filled in.

Guided completion continues to select the next required field until the user has completed your Goal form.

After a form is completed, users can edit any unprotected field. Guided completion is inactive because no more fields require information.

Restoring guided completion

During form completion, the user can interrupt guided completion by selecting another of your application's forms as the Goal form. After a user selects a new Goal form, ObjectVision ignores guided completion and determines a field sequence for the selected form.

File | Resume quickly returns users to guided completion.

This user-selected form continues to be the Goal form until the user closes it, selects a new Goal form, or restores the previous field sequence (with File | Resume). When the user closes a Goal form, ObjectVision selects the previous Goal form as the new Goal form.

Minimal recalculation

Whenever a user enters a new value or changes an existing value in your application, only those values in your application that are based on that value are recalculated. Recalculation occurs

whether the user is guided by the field sequence or is manually selecting fields.

Minimal recalculation lets users perform "what if" analyses within your application, just as they might within a spreadsheet. For example, recalculation can determine the effect of changing a field's value or show the results of other options for a field.

When fields need recalculation, ObjectVision first removes the current value of any field that is based on the changed value and then displays the new value in all dependent fields.

After a user changes a value, ObjectVision might be unable to immediately recalculate a value for all dependent fields. When a changed value requires following other branches in your decision tree, new values might be required. ObjectVision automatically selects any fields requiring new values.

Field | Calculate recalculates a value for a calculated field.

Users can choose the Field | Calculate command to recalculate any field that lacks its calculated value. After a user chooses this command, ObjectVision determines whether more information is needed to recalculate the value.

Decision trees

You use the Tree Tool to create *decision trees* that define procedures for calculating field values. Some procedures might be simple mathematical operations such as adding a column of numbers. Other procedures might be more complex and involve examining a variety of conditions before calculating a value.

For example, the amount of temporary insurance granted in an insurance application might be determined by a procedure that considers the applicant's age and health and the type and amount of insurance requested.

Spreadsheet formulas are similar to decision trees, and you can use the Quattro Pro-compatible functions described in Chapter 8, "@Function commands," to create decision trees. You can also create decision trees to represent conditional logic, as well as mathematical formulas.

Figure 1.4
Decision tree for the Credit Terms field

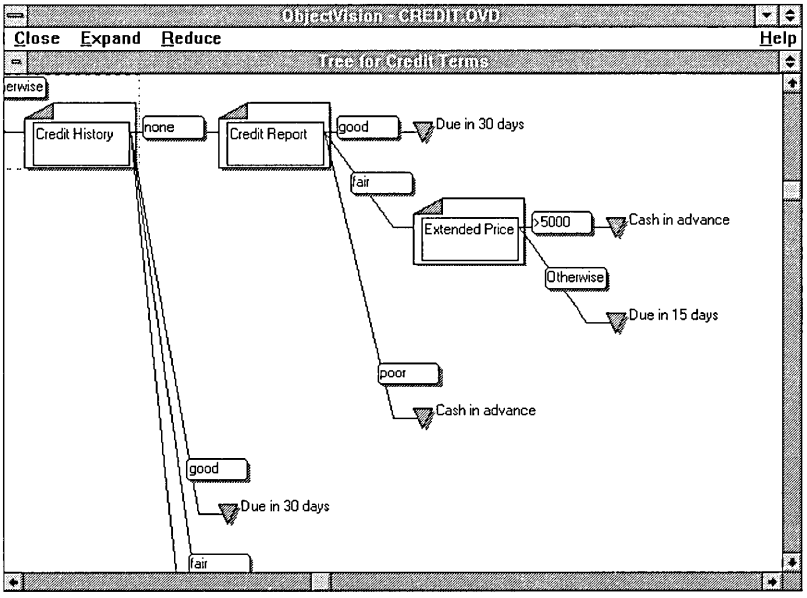


Figure 1.4 shows the Credit Terms decision tree for the *Credit* sample application. This field calculates credit terms for a customer based on the customer's credit history and order size.

Decision trees can define a complex decision-making process by breaking it into a series of smaller steps. At each step, a single element of the tree, called a *node*, is evaluated.

Decision trees calculate fields automatically.

As the user fills in your form's required field values, ObjectVision calculates any decision-tree values dependent on the user-entered values.

If you are a spreadsheet user, you are familiar with this concept. It's like a spreadsheet cell that has a formula associated with it.

Your decision trees return the value defined by the conclusion node.

When you create a decision tree, you define all anticipated field values as branches in your tree. Each *branch* can define other branches based on another field value in order to define complex relationships between various decision criteria.

When evaluating a decision tree, ObjectVision follows the branch appropriate to the current field value. When ObjectVision reaches the last branch node, the decision tree defines a conclusion that results in a calculated field value.

When ObjectVision determines which of your tree branches to follow, the actual field value is evaluated.

The tree nodes, or boxes, contain field names for other fields in the application. When a node is on the current *decision path* in a decision tree, that node is used to calculate the selected field's value.

Your users can perform "what if" analyses by simply changing the value of a field. ObjectVision recalculates all of your application's decision trees that contain references to that changed field.

Tree Tool

You use the Tree Tool to create and edit decision trees for any field in your application. The Tree Tool's cut, copy, and paste operations let you manipulate your decision trees graphically.

Whenever you create a new decision tree for a field or modify an existing tree, you also change the decision-making logic associated with that field.

Expressions

Within decision trees, you use *expressions* to compute a value based on a combination of other values. For example, you might need to multiply a series of numbers. You can use ObjectVision expressions that are similar in concept, use, and syntax to Quattro Pro's formulas.

You can use ObjectVision expressions for decision tree conclusions and branch conditions.

When you use expressions in your decision tree conclusion nodes, your expression can contain a constant value, such as *Yes* or *No*. Your expressions can also contain operators or calculate a value based on other field values.

A decision tree without branches has only one conclusion node and always returns the same value, similar to a formula in a spreadsheet cell.

Each branch of a decision tree contains an expression that determines when a branch in the tree is used. A branch expression can be either a constant value or a formula that computes a value. A constant value, such as *Excellent*, *Good*, or *Fair* can be used as a branch condition, as can a formula, such as *">= Total"*.

Expressions can be a maximum of 4,096 characters.

The syntax and use of ObjectVision expressions is compatible with Quattro Pro formulas. An expression can contain combinations of constant values, references to field values, mathematical and comparison operators (such as + and >=), and @functions (such as @INT). Like field values, expressions can contain a maximum of 4,096 characters.

Conditions and Conclusions

When you use the Tree Tool to create a decision tree, you can use the Condition dialog box or the Conclusion dialog box to create and modify expressions. You can type or paste @function names and field names into the expression as you edit it. When you finish editing an expression, the validity of the expression is checked.

External links

Your applications can use *external links* to data as an alternative to requiring user input or computing a value in a decision tree. You can use ObjectVision's Link Tool to create, modify, or delete external links that read from and write to data files created by other applications.

You can use ObjectVision's external links in your applications to link with these data file types:

- ASCII
- Paradox
- dBASE-compatible
- Btrieve
- Dynamic Data Exchange (DDE) to other Microsoft Windows applications

You can create all five types of links by choosing Tools | Links. When the Links dialog box appears, specify the type of link you want, where the external data is, and which ObjectVision fields to connect.

ObjectVision also provides a set of @functions for links you can use in your decision trees. Each data file type has an @function for opening that specific file type. For example, to open an ASCII file

for reading or writing, you use the @ASCIIOPEN link function. To open a Paradox file for reading or writing, you use the @PXOPEN link function. For more information, see Chapter 8, "@Function commands," and Chapter 10, "Linking to data files."

When your applications link to external data, the operations your links perform observe the conventions of the application that created the data file. For example, when a user searches for an existing field value using inexact match, and a match for that pattern is unavailable, Paradox returns the closest matching record and dBASE returns nothing.

ASCII files

Your ObjectVision applications can exchange data with ASCII files. Many popular spreadsheet, database, and word-processing applications read and write ASCII files. You can also use ASCII files for transferring data by modem to a remote location.

Your ObjectVision applications can read data from ASCII files and connect it to fields in your forms. You can also write any of your form's field values into a linked ASCII file.

Paradox

Your ObjectVision applications make it easy for users to get existing data from Paradox files. The links you create can also either update your application's linked field values or update the database's records.

If you create a Paradox table using ObjectVision, an index is automatically created on the first field in your form. You can put the field you want indexed at the top of the form and reposition it after the table is created.

Your applications can be used in a multi-user environment, where data integrity is essential. All users can use your ObjectVision application to read linked Paradox records. ObjectVision performs record locking and notifies a user who tries to save revised values if another user has already updated the same record during the current session.

dBASE

Your ObjectVision applications make it easy for users to get existing data from dBASE-compatible files. The links you create can also either append your application's linked field values or update the database's records.

Your applications can be used in a multi-user environment, where data integrity is essential. All users can use your ObjectVision application to read linked dBASE records. ObjectVision performs record locking and notifies a user who tries to save revised values if another user has already updated the same record during the current session.

Btrieve

Your ObjectVision applications make it easy for users to get existing data from Btrieve files. If you are linking to existing Btrieve data, you *must* use Novell's XQL application to create the data dictionary files. The links you create can also either append your application's linked field values or update the database's records.

Your applications can be used in a multi-user environment, where data integrity is essential. All users can use your ObjectVision application to read linked Btrieve records. ObjectVision performs record locking and notifies a user who tries to save revised values if another user has already updated the same record during the current session.

DDE

Windows provides the Dynamic Data Exchange (DDE) for transferring information between applications.

You can create DDE links from your application to other Windows applications such as Excel and Word for Windows.

DDE links have the advantage of transferring data automatically whenever the information source file receives a new value. With DDE links, you can seamlessly integrate your application's data with files from a variety of Windows products. You can also link your application to multiple ObjectVision applications when you need to create an application larger than the ObjectVision system limits.

C H A P T E R

2

Getting started

Before you begin working with ObjectVision, you need to

- check the contents of your ObjectVision package
- make sure you have the correct equipment and operating system for running the program
- run the installation program
- read the README file
- start ObjectVision from within Windows

This chapter discusses each of these requirements and procedures. When you finish this chapter, you can go to work with ObjectVision. If you are unfamiliar with Windows, you might want to read the brief introduction to using Windows also included in this chapter.

Supported equipment

The next two sections cover both necessary and optional equipment supported by ObjectVision.

Necessary equipment

Here's what you'll need to run ObjectVision:

- **Computer.** ObjectVision is designed to run on IBM AT, PS/2, and fully-compatible computers using the Intel 286, 386, or higher processor.
- **Operating System.** ObjectVision runs with Microsoft Windows 3.0, which requires DOS 3.1 or later. To use Paradox links, you must run Windows in either 386 enhanced mode or standard mode. Real mode is not supported by the Paradox links.
- **Memory.** To run ObjectVision, you'll need at least 640K of random-access memory (RAM); we strongly recommend adding a minimum of 256K of extended memory.
- **Disk Drive.** Your computer must have a hard disk with at least approximately 1MB of free disk space.
- **Graphics Card.** You need a graphics card and a display capable of high-resolution graphics. ObjectVision supports the following graphics cards (or any fully-compatible cards):
  - IBM Color/Graphics Adapter (CGA)
  - Hercules (monochrome) Graphics Card
  - IBM Enhanced Graphics Adapter (EGA) (color or monochrome)
  - IBM Video Graphics Array (VGA) (color or monochrome)
  - IBM 8514/a Graphics Adapter
  - IBM Multi-color Graphics Array (MCGA) (PS/2 Model 30)
  - Olivetti/AT&T (PVC or monochrome)
  - Olivetti OEC
  - AT&T VDC750
  - Compaq Portable 386 Plasma
  - Video 7 VGA

Optional equipment

The following equipment is not necessary to run ObjectVision, but will enhance its performance and your ease of use:

- **Extended memory card.** You can add extended memory cards to your PC to improve Window's performance on your system. The more memory your system has, the better:

- To run Windows in 386 enhanced mode on your 80386 or 80486 PC, you need at least 2MB of RAM, 1,024K of which is extended memory.
- To run Windows in standard mode on your 80286 PC, you need at least 256K of extended memory.

■ **Mouse.** We strongly recommend using a mouse with ObjectVision. Any pointing device fully compatible with one of the following will work:

- Hewlett-Packard mouse
- IBM PS/2 mouse
- Logitech mouse
- Microsoft mouse (bus or serial)
- Mouse systems mouse (on COM1 or COM2)
- Olivetti mouse
- Olivetti/AT&T keyboard mouse

■ **Printer.** You can use any printer that is supported by Windows.

Installing ObjectVision

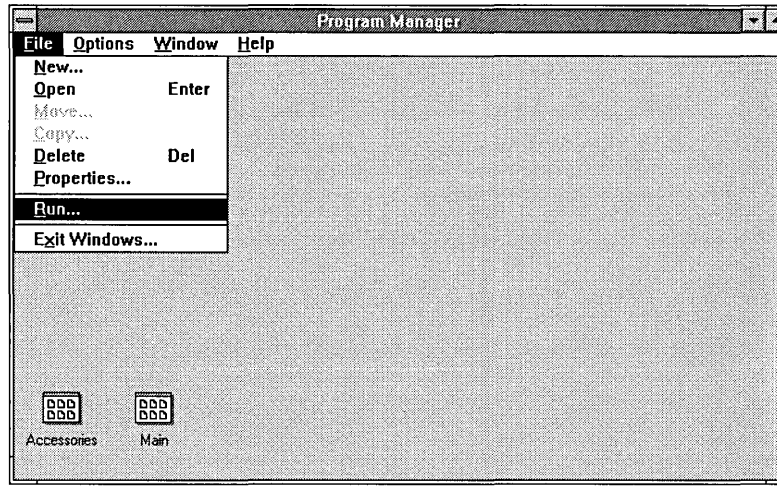
The first step to getting ObjectVision up and running is to run the installation program. The installation program performs the following operations:

- creates a directory on your hard disk for the ObjectVision program and sample application files.
- copies ObjectVision program files and sample application files to the newly created directory.
- adds an ObjectVision Group to the Program Manager.
- modifies your WIN.INI Windows initialization file so that ObjectVision will run properly under Windows.

To run the installation program and install ObjectVision, follow these steps:

1. Start Windows.
2. Put the ObjectVision disk in drive A.
3. Choose Run from the Program Manager File menu as shown in Figure 2.1.

Figure 2.1
Program Manager File I Run command



4. In the Run window Command Line field, type the following line:

   ```
   A:\INSTALL
   ```

   and then choose OK.

Figure 2.2
File I Run dialog box

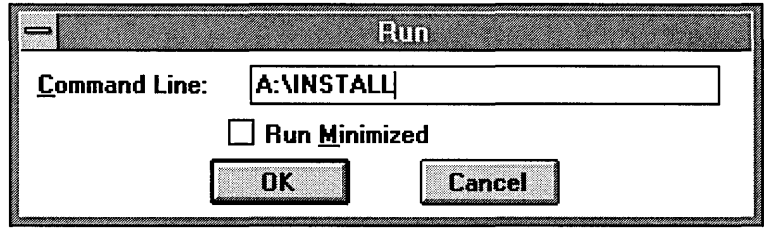
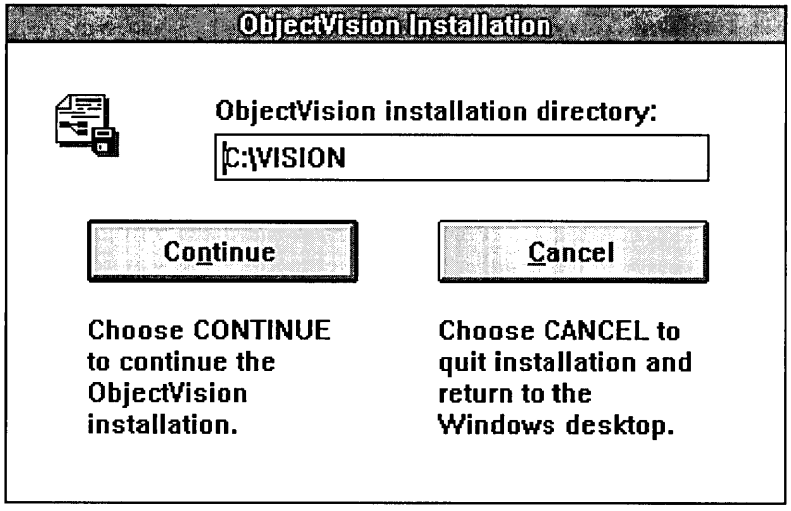


Figure 2.3
First installation window



Press *Tab* to move between fields in the installation dialog box. If you need to make changes to the information you type in, press *Shift+Tab* to step back through the fields.

The default drive and directory where ObjectVision is installed is C:\VISION. You can change this to any drive and directory combination you want.

5. When you have filled in the directory name you want ObjectVision installed in, press *Enter* or choose Continue.

   The installation program displays a window asking you to confirm installation in the specified directory.

6. If the drive and directory are correct, choose Continue. If they are not correct, choose Cancel.

   The installation program now copies all the necessary files to the specified Install directory. During the copy operation, the name of the file currently being copied displays. Note that the Cancel Installation button remains active. If you cancel the installation in progress, the Install directory will contain a partial set of unusable files that you must delete from your hard disk.

After all the files have been copied, the installation program informs you that installation is complete.

At this point, you can choose to view the ObjectVision README.TXT file or to return directly to the Windows desktop. Although viewing README.TXT is not a required part of ObjectVision installation, you should read README.TXT before you start to use ObjectVision. This file includes information developed too late to be included in the printed documentation.

7. Choose View README now, or choose Cancel README to return directly to the Windows desktop.

   Once you complete ObjectVision installation, you can start ObjectVision without restarting windows.

Using the Paradox Engine

If you plan to use Paradox links within ObjectVision, you might need to setup the Paradox Engine as described in Appendix D, "Configuring the Paradox Engine." If you are simply running multiple instances of ObjectVision locally, or you aren't using Paradox links, you won't need to configure the Paradox Engine.

Be sure to configure the Paradox Engine if:

- You are running ObjectVision concurrently with Paradox in a DOS box
- You are using Paradox links to tables shared on a network
- You want to change the default engine resource limits or network configuration.

Moving the ObjectVision icon

The installation program creates an ObjectVision Group in Windows and places the ObjectVision icon in that group. If for some reason you want to move the ObjectVision icon to another group, you can do so. Moving the icon is easier if you use a mouse, but a mouse is not required.

Using a mouse to move the icon

To move the ObjectVision icon to another group and delete the ObjectVision Group, follow these steps:

1. Position the ObjectVision Group and the other group (Windows Applications, for example) so that both the ObjectVision icon and some part of the other group's window are visible.
2. Drag the ObjectVision icon into the other group's window. Click anywhere in the empty ObjectVision Group to select it.
3. Choose Delete from the Program Manager's File menu. Windows asks you if you want to delete the ObjectVision Group.
4. Choose Yes.

Using keys to move the icon

1. Use *Ctrl+F6* or *Ctrl+Tab* to select the group window in which you want the icon to appear.
2. Choose File | New.
3. In the New Program Object dialog box, choose Program Item.
4. In the Program Item Properties dialog box, in the Description field, type the name that you want to appear below the icon (`ObjectVision`, for example). Then press *Tab* to move to the Command Line field.
5. In the Command Line field, type the full path to the executable program file VISION.EXE (for example, `C:\VISION\VISION`). Then choose OK.

Now the ObjectVision icon is in the group where you want it. To remove the icon from the ObjectVision Group and delete the empty group, follow these steps:

1. Select the ObjectVision Group. The ObjectVision icon (the only icon in the group) should already be selected.
2. Choose File | Delete. Windows asks you if you want to delete the ObjectVision item.
3. Choose Yes. Now the ObjectVision group is empty.

4. Choose File | Delete again. Windows asks you if you want to delete the ObjectVision Group.
5. Choose Yes.

Starting ObjectVision

You can start ObjectVision from the Windows desktop or from the DOS prompt.

From the Windows desktop

To start ObjectVision from the Windows desktop, follow these steps:

1. Start Windows.
2. Select the group window that contains the ObjectVision icon.
3. Double click the icon. If you aren't using a mouse, select the icon with the arrow keys, and then press *Enter*.

From the DOS prompt

Follow these steps to start ObjectVision from the DOS prompt:

1. Change to the directory where you installed ObjectVision.
2. Type

   ```
   cd \vision
   ```

   and press *Enter*.
3. To start ObjectVision and Windows at the same time, type

   ```
   win vision
   ```

   and press *Enter*.
4. To start ObjectVision and Windows, and also open an ObjectVision application from the DOS prompt, type this command:

   `win vision` *application-name*

   and press *Enter*. For *application-name*, you type the name of the ObjectVision application file you want to open, omitting the .OVD extension. For example, this command opens the *W4form* sample application when you start ObjectVision from DOS:

   `win vision` *W4form*

Modifying your WIN.INI file

The ObjectVision installation program makes some modifications to the WIN.INI file in your Windows directory. In most cases, it is not necessary for you to make any further modifications to that file.

There are some circumstances, however, under which you might want to modify the WIN.INI file further. You can use the Windows Notepad, the Windows Write application, or another text editor of your choice to modify the WIN.INI file.

The load and run statements

Your WIN.INI file contains both a load statement and a run statement. By default, these two statements are blank and look like this:

```
load=

run=
```

If you want ObjectVision to be loaded (but not run) every time you start Windows, edit the load statement to read:

```
load=C:\vision\vision.exe
```

If you want ObjectVision to be loaded **and** run every time you start Windows, edit the run statement to read:

```
run=C:\vision\vision.exe
```

Using Windows

If you are already familiar with other Windows applications, you can skip this section. If you are new to Windows, read this section for an introduction to the Windows environment.

For a complete listing of ObjectVision keyboard operations and their mouse equivalents, see Appendix A, "Keyboard and mouse operations."

For a more comprehensive explanation of Windows, see the Windows documentation that you received when you purchased Windows.

Application windows

Figure 2.4 illustrates the ObjectVision application window. The following paragraphs describe each part of the display:

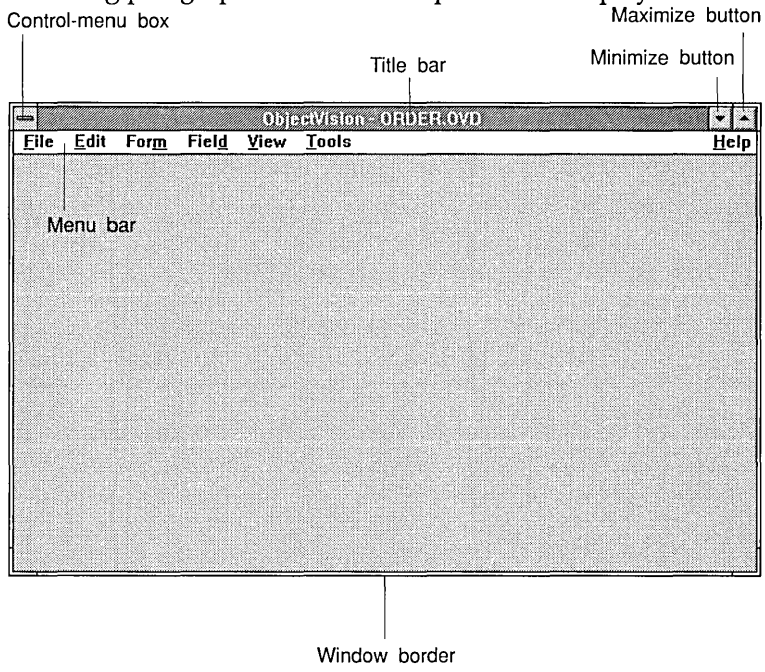
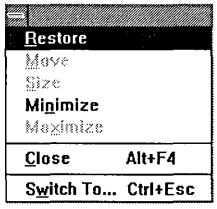


Figure 2.4
ObjectVision window

Control-menu box



The Control-menu box, in the upper left corner of each window, opens the Control menu, where you choose commands common to all Windows applications.

You can use the Control menu commands to move, resize, and close the application window using the keyboard. You can also use the Control-menu box to switch to the Program Manager or any program loaded on your desktop.

To open the Control menu, you can either click the Control-menu box with the mouse or press *Alt+Spacebar*. After you open the Control menu, you can choose a menu command using the techniques described in the "Choosing menu commands" section later in this chapter.

Title bar

The title bar shows the name of the application, ObjectVision, and the name of your open file, if any. The title bar displays `(Untitled)` after the application name if no file is open, or if the current file has not been saved.

Each ObjectVision application file you open also displays in a window with its own title bar containing the form name and the form status.

When more than one window is open on your desktop, the window you are currently working in is called the *active* window. The active window always appears in the foreground, and its title bar is highlighted with a different color or intensity.

When you want to work in another window, you select it by

- clicking any visible portion of an inactive window
- pressing *Alt+Tab* repeatedly until the window you want is active
- pressing *Ctrl+Esc* to display the Task list

You can switch to any application displayed in the Task List dialog box.

You can also move a window by placing the pointer in the title bar and dragging the window.

Menu bar

The menu bar lists ObjectVision's menu names. You use menus by selecting a menu to open it and then choosing an available command from that menu.

When a menu command is inappropriate in the current context, the command name is dimmed to indicate that it is unavailable.

In ObjectVision, you see a different set of menus when you are working with the Tree Tool, the Form Tool, or the Stack Tool; but you choose all menu commands in the same way. Mouse and keyboard methods for choosing commands are explained in the "Choosing and canceling" section later in this chapter.

Maximize, Minimize, and Restore buttons

The Maximize and Minimize buttons are in the upper right corner of the application window. The Restore button replaces the Maximize button on a full-screen window.

When you choose the Maximize button, the window enlarges and displays full-screen. Maximizing is very useful when you want the application to display a large amount of information.

When you choose the Minimize box, the window shrinks down to an icon on your desktop. Minimizing leaves the application loaded in your system's memory so you can open it instantly when you want to use it again.

You can use the Control | Restore command to return an icon to its previous size and position, or the Control | Maximize command to maximize the icon as a full-screen window.

After you choose the Maximize button, it changes to the Restore button. When you choose the Restore button, the window returns to its previous size and position.

Window border

Using a mouse, you can drag a border to change only one side or a corner to change two sides at a time.

When a window occupies less than the full screen, you can resize it by moving its borders. You can change the size and shape of most windows on your desktop.

You can also choose the Control | Size command to resize and reshape a window. After you choose this command, the pointer changes to a four-headed arrow, and you press an arrow key to select the border you want to move. You press the arrow key to move the border, and a dotted outline indicates the new size and shape as you change it. When you finish, press *Enter*.

Choosing and canceling

All Windows applications have a common Control menu and a set of unique menus in a menu bar. Commands on ObjectVision menus can be actions the application carries out or characteristics you assign to objects such as text or fields.

You select a menu name to open a menu, and then you choose an available command from that open menu. If you decide against choosing a highlighted command, you can cancel a menu.

Choosing menu commands

To choose a command using the mouse, position the pointer on the menu you want to open, hold down the mouse button, drag the selection pointer to a command name, and then release the mouse button.

You can press *Alt* or *F10* to select the menu bar, press ← or → to select the menu you want, and then press *Enter* to open the selected menu.

As a keyboard shortcut, you can move to the menu bar, select the menu you want, and open it in one step. Press *Alt* and then press the underlined letter in the menu name. For example, to open the File menu in ObjectVision, you can press *Alt+F*.

To choose a command from an open menu, you move the selection pointer by pressing ↓ or ↑, and then pressing *Enter*.

You can choose some application commands using shortcut keys, without opening a menu. The shortcut keys appear on the menu, after the command names. For example, you can get ObjectVision Help by pressing *F1*.

Canceling menu commands

If you decide to cancel a selected command, you can click anywhere outside an open menu.

You can cancel an open menu by pressing *Alt*, *F10*, or *Esc*. When you press *Alt* or *F10*, you cancel the menu and return to the application workspace. When you press *Esc*, you cancel the menu, but the menu bar is still selected so you can select another menu name.

Using dialog boxes

When you see an ellipis (...) after a command name, it indicates that a dialog box appears after you select that command. Dialog boxes request information from you and display additional messages and warnings.

You can type information into a dialog box, select options, or select items from a list. When you finish, you choose a command button to either carry out the action or cancel it.

Setting dialog box options

As is the case when choosing menu commands, you can use a variety of keys to set dialog box options.

With the mouse, you can click an option to select it.

When dialog-box option names have an underlined letter, you can press *Alt* and the underlined letter to move the selection pointer to that option. Or, you can press *Tab* to move the selection pointer from left to right and top to bottom in a dialog box. When you

press *Shift+Tab*, the selection pointer is moved back to the previous option.

If your selection pointer is positioned on a check box or a radio button, you can press *Spacebar* to select that option. When you press *Spacebar*, the selection toggles between check and uncheck. For example, if you move the selection pointer to a check box that is already selected and press *Spacebar*, the box is unchecked. If the check box is unchecked before you press *Spacebar*, the option is checked.

Editing text boxes

You can use standard Windows text-editing techniques in dialog boxes. Using the mouse, you can double-click any letter to select an entire word, or drag to select text.

When you press *Home*, the text cursor moves in front of the first character. When you press *End*, the text pointer moves to the space immediately after the last character. To remove characters, press *Del* to erase a character to the right of the pointer or *Backspace* to delete a character to the left of the pointer.

Choosing command buttons

After you finish selecting options in a dialog box, you choose command buttons to carry out the specified actions (or apply the specified attributes), or to cancel them.

Using a mouse, you can click OK or Cancel to choose a command button.

You can also position the selection pointer on the command button you want by tabbing to it, and then pressing *Spacebar* or *Enter* to choose the button and carry out the command.

C H A P T E R

3

Using an ObjectVision application

This chapter explains how users run your ObjectVision applications under Windows. It is intended to help you understand how to design your ObjectVision applications.

To use your ObjectVision application, users start ObjectVision or ObjectVision Runtime and select the file they want. Users complete the application by filling in fields in your application's form or forms. This process is called *form completion.*

Users fill in your application's fields in much the same way they would type information into any form, except that your application does much of the work for them. Your ObjectVision applications help users by automatically selecting only the fields they need to fill in.

Another ObjectVision feature can help users understand how your application calculates values. Users can display the decision trees associated with your calculated fields to see a visual representation of the logic that gives the field a value.

For example, if a field in a life insurance form indicates temporary insurance is unavailable, the user could display your decision tree and understand that temporary insurance is unavailable for policies as large as the one the user specified.

Opening an application

When users start ObjectVision without also starting an application, the ObjectVision window appears with a blank workspace. From within ObjectVision, users open your ObjectVision application by choosing the File | Open command.

Users can choose File | Open to open a different application when they are in form completion mode. ObjectVision allows only one application to be active at a time, so when users choose File | Open, ObjectVision closes the application they were working on.

If any values in the fields were changed, ObjectVision prompts the user to save the changes before closing the current application. After the user responds, the Open dialog box appears.

Finding an application

The File | Open dialog box lists those files in the current directory that have a file-name extension of .OVD. These are assumed to be ObjectVision application files. Users can select and open a file from this list by double-clicking it.

Using the keyboard, a user can select a file name by tabbing to the list, typing the first letter of the file name, and then choosing OK or pressing *Enter*. If there is more than one file name with the same first letter, a user can continue pressing the first letter to select another file name.

If the file a user wants is in a different directory or has a different file extension, they can specify different directories or file-name extensions.

When the application file name is not listed in the File | Open dialog box, users can type the complete path name of the file (including its extension, if it is not .OVD).

Users can also type a file specification, such as `\path\*.ext`, and then press *Enter* to list matching file names in the specified path. Another drive or directory can also be selected from the list to display matching file names on that drive or in that directory.

For example, suppose the application you want, HIRE.OVD, isn't listed in the File | Open dialog box. Your current directory is WINDOWS, and you know that HIRE.OVD is in the directory

FORMS. You can select the default file name in the File Name text box, delete it, and then type `\forms\hire`.

Or suppose you plan to work with several applications in the FORMS directory. You can enter `\forms\*.OVD` in the file listing box, and a list of files in the FORMS directory with the extension .OVD appears.

When a user opens your ObjectVision application files, the Goal form appears with its first required field selected. You specify the Goal form in the Stack Tool, where you define the order of the forms in your application.

Saving changes

To save changes you have made to an application, or to save data entered into a form, you can select either the File | Save command or the File | Save As command:

- The Save command saves the changes to the current application file name, overwriting the previous data. If the current work is unnamed, the File | Save As dialog box appears.
- After you choose the Save As command, the Save As dialog box appears so you can enter a name for the new file. The current application is saved to the file you name, leaving the original application as it was. If you name a file that already exists, ObjectVision asks whether you want to replace the file. The Save As command also changes the name of the application you are working with, and the next time you choose File | Save the new file name is used.

Discarding values

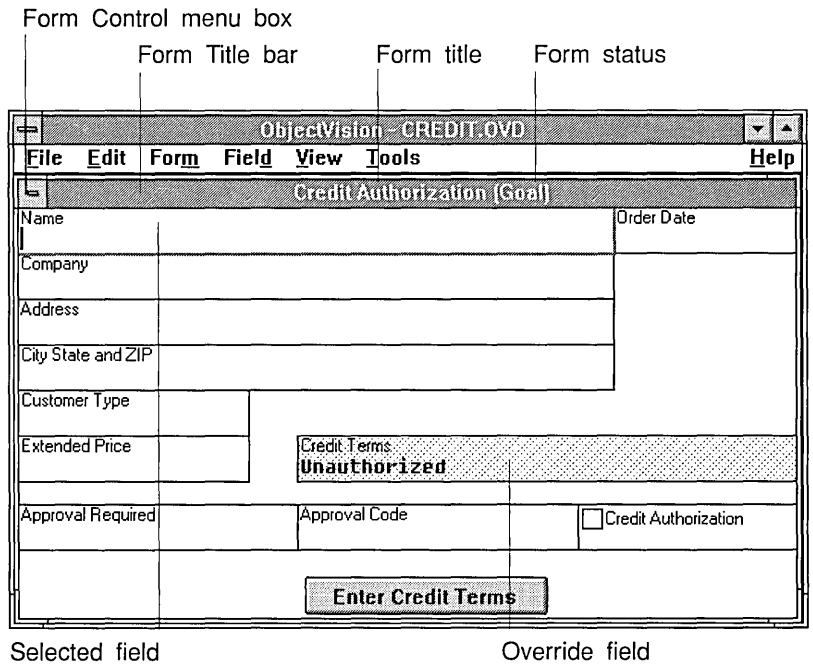
When users decide to discard the values they entered into your application's forms, they can choose either the Edit | Clear All or the Form | Clear command. The Edit | Clear All command erases all the current field values they entered, and the Form | Clear command erases *only* the current values in the active form.

When users save and close an application after clearing the field values, the empty forms are stored for future use.

Identifying parts of a form

Figure 3.1 illustrates the parts of a typical form. The paragraphs that follow Figure 3.1 describe each form component individually.

Figure 3.1
Parts of a typical form



Understanding form status

The application title bar displays the form name and its status. The form status appears enclosed in parentheses following the form name and indicates one of the following values:

- *Goal*: A user is filling in the selected field in a form that has one or more fields requiring values. The top form in your stack of application forms is automatically selected as the Goal form. When a user selects another form, that form becomes the Goal.
- *Prompt*: A temporary form or scratchpad is gathering information required to determine a value on a Goal form. ObjectVision dismisses Prompt forms once users enter the required information.
- *Complete*: All fields on the form have a value. Users can change the value of any unprotected field.

- *Untitled*: The current application has not been saved.

When more than one form displays on the workspace, only the frontmost form is the *active form* and can receive a user's keyboard input. The title bar of the active form is always highlighted with a different intensity or color. You can select another form to receive keyboard input by clicking any visible part of the form or by selecting the form in the Form | Select dialog box.

If the form is smaller than the full-screen, you can move the window to a new location by dragging the Title bar or by choosing the Move command from the Control menu.

Selecting fields

If any fields requiring user input on the active form are visible, one of those fields is the *selected field*. A selected field is the active field on a form which displays everything a user types. In Figure 3.1, the selected field is Name.

A *calculated* field displays a heavy outline with a dashed line inside it when it is selected. A calculated field gets its value either from a decision tree or from an external link.

A field requiring user input displays only a heavy outline around that field's border when it is selected.

Overriding values

An *override* occurs when users alter information in a calculated field. To indicate that a field does not contain its calculated value, overridden fields display a dot pattern. In Figure 3.1, Credit Terms is an override field.

When a user wants an overridden field to be returned to its calculated value, they use the Field | Calculate command.

Control-menu box

The Control-menu box to the left of the form title is similar to the Control-menu box to the left of the ObjectVision Title bar, but has its own shortcut. To open the form window's Control menu, click the Control-menu box or press *Alt*+- (hyphen).

The Control menu icons are slightly different to remind you of their different shortcuts. The application window Control menu

icon is a long bar (*Alt+Spacebar*) and the Form Tool window Control menu icon is a short bar (*Alt+-* (hyphen)).

Form border

The form border displays only around forms larger than the application window. You can't use the form border to change the size of the form as you can for application windows. However, you can resize Scratchpad forms, which ObjectVision uses to prompt for values that are not contained on a form.

Guided form completion

A unique ObjectVision feature is *guided completion*, which makes it easy for users to fill in forms. ObjectVision determines the field sequence for required information on the Goal form, based on which fields require user input. Calculated fields and fields read from external sources are omitted from this selection of fields.

Users can let ObjectVision select the next required field and form according to this field sequence and ignore all other fields and forms. Users can also interrupt the field sequence to select any form or field manually. If a user does interrupt the field sequence, they can return to it by choosing the File | Resume command.

Selecting forms

To select another form, the user chooses Form | Select. The Form | Select dialog box lists form titles for the current application according to their order in the application's stack of forms. Once a form is selected, it temporarily appears in front of any other visible forms and also moves to the top of the form stack.

Whenever a form is selected, that form becomes the new Goal form unless it is already complete.

ObjectVision suspends processing the previous Goal form, determines the field sequence for the new form, and then selects the first field on the new form requiring user input. A user can resume filling in the previous Goal form by closing the active form, selecting the previous Goal form, and choosing the File | Resume command.

Selecting fields

ObjectVision's guided completion selects fields requiring completion starting from the top left corner of a Goal form and moving toward its bottom right corner. A field's bottom right corner determines its placement in the field sequence order.

For example, in Figure 3.1, the Credit Authorization field comes after the Approval field because its bottom right corner is on the same baseline as the Approval field, and it is to the right of the Approval field.

Once a user completes the Goal form, pressing *Enter* simply moves from field to field, skipping calculated fields. Users can edit any of the unprotected values in a form.

After users complete a session, they can save the form's data to a linked data file. See Chapter 10, "Linking to data files," for complete information about linking to external data files. A user can choose the File | New command or the Form | Clear All command to begin a new data entry session in a blank form.

Moving between fields

When users finish typing a field value or selecting options in a field, they can press *Enter* or *Tab* to enter that value into the field. They can also click any portion of another field to select that other field and enter a value.

After pressing *Enter*, guided completion selects the next field, which can be on the active form or a different form in the application. When the form status is Complete, the next field on the active form is selected so users can edit its value.

When a user presses *Tab* to enter a typed value, the next field on the active form is selected, whether it requires a value or not. The next field is determined only by the physical layout of the fields, going from left to right and from top to bottom. Clicking another field interrupts guided completion, just as pressing *Tab* does.

To erase a value that has been typed but not yet entered, the user can press *Esc*. If the user has already gone on to the next field, the user can choose Edit | Undo or press *Alt+Backspace* immediately after pressing *Enter* to back up to the previous field and erase its value.

Determining values

Values for calculated fields are computed as soon as a user enters all required information. Additionally, when a calculated field is selected, the Calculate command is available on the Field menu.

Users can choose Field | Calculate to have ObjectVision compute the value of the selected calculated field. ObjectVision then selects any fields required for computing that field's value.

If a user has previously overridden a field's calculated value, Field | Calculate restores the calculated value.

Selecting another field

Users can use the Field | Find command to manually select another field to receive input. The Find dialog box lists the current application's field names in alphabetical order. After a user selects a field from the list and chooses OK, the form containing that field appears in front of any other displayed forms.

When more than one form contains the selected field, the form closest to the top of the stack appears in front of any other displayed forms. This newly selected form becomes the new Goal form just as if a user selected it using the Form | Select command.

If the field you select is not contained on any form, the Scratchpad form displays the field.

Editing field values

Pressing *Enter* always enters a value in the active field. It might enter an empty text string or select an option from a selection list, for example. Users can click another field to select it, or tab to another field if they want to leave a field empty.

Users can use the keyboard or the mouse to edit values they have typed or entered while filling in your application's forms. The following paragraphs describe text selection and editing methods.

Pointer positioning

With a mouse, a user can click anywhere in field value text to reposition the pointer. Double-clicking a letter selects the entire word. Dragging is also a fast way to select text.

Users can also press *Home* or *End* to position the text pointer in a field. Pressing *Home* moves the pointer to the left of the first character in the field. Pressing *End* moves the pointer to the right of the last character in the field.

Text editing

When a user drags to select text, the next characters typed replace the selected text.

With the keyboard, users can press *Backspace* to erase characters to the left of the pointer. Pressing *Del* removes characters to the right of the pointer.

If a user has typed a new value, but has not yet pressed *Enter*, pressing *Esc* removes the new value and restores the field to its previous value. If a user has typed a new value and then pressed *Enter*, but no editing changes have been made since entering the value, the user can choose the Edit | Undo command or press *Alt+Backspace* to restore the previous value.

Error values

Various conditions can cause an error value to appear in a field. The error messages a user is most likely to see while filling out a form are

- **ERR:** indicates that an error has occurred while evaluating an expression associated with that field. For example, dividing a value by zero causes this error.
- **NA:** indicates that the value entered is an unanticipated condition in the field's decision tree. Usually, adding the Otherwise conclusion takes care of unanticipated conditions. However, NA might also appear if the DDE link you specify is established, but the DataField argument is unrecognized by the Windows application.

In general, errors can only be cleared by changing the value of other fields used in the calculation, or by changing the decision tree logic associated with the field. Viewing the decision tree for the field is often useful in determining the cause of the error.

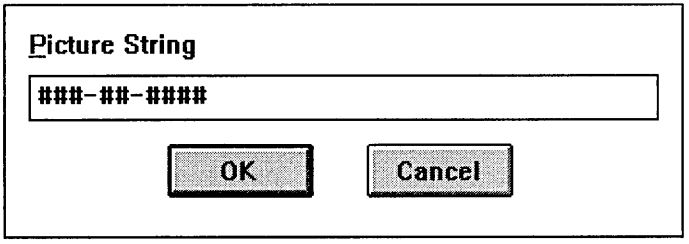
A user might also see a field display a value as a series of pound signs filling the field (#######). This is not an error, but it indicates that the calculated value is larger than the field can display.

Understanding field types

There are several different types of ObjectVision fields. The way users enter data depends on the field type:

- **Text/numeric:** Users can enter text or numbers in a text/numeric field. For example, users might enter names and addresses in text/numeric fields.
- **Selection List:** Users select one option from a list of possible values in a list field. For example, users might select Apartment, Duplex, or Single Unit from a Home Type list field.
- **Check box:** Like a list field, a check box field displays a list of options from which users select a value. In the check box field, however, each option has a check box next to it.
- **True/false:** A true/false field is like a check box field, except it contains a single box. A checked box means that the value is true; an empty box means that the value is false. A field that said U.S. Citizen, for example, might be a true/false field.
- **Scrolling:** Unlike other field types, the text typed into a scrolling field can exceed the size of the field. Text typed into a scrolling field automatically wraps and can be scrolled. A scroll bar displays on this field type only when the field is selected.
- **Button:** A button field, like a true/false field, has either a true or false value. A button field has a value of false until a user activates it and then the field has a value of true. For example, the *Order* sample application has a Save to database button at the bottom of Order form. When a user activates that button, a True value is evaluated in the button's decision tree, and the @STORE function writes the current data to the ORDER.DB database file.
- **Picture:** A picture field controls values users type into a field during data entry and also eliminates typing default or repetitive values. For example, a Social Security number field might have a picture defining ###-##-#### as the pattern for the value, as shown in Figure 3.2.

Figure 3.2
Field Type I Picture dialog box



The paragraphs that follow explain data entry for each type of field.

Text/Numeric fields

Text/numeric fields are similar to blanks in a paper form. You can type text or numeric values in these fields. For example, you might enter a date, a name, or an address in a text/numeric field.

Anything you type is inserted at the current pointer location. You can reposition the pointer using the *Home, End,* ←, and → keys:

- *Home* moves the pointer to the left of the first character in the field.
- *End* moves the pointer to the right of the last character in the field.
- ← moves the pointer one character to the left.
- → moves the pointer one character to the right.

You can also reposition the pointer by clicking the mouse where you want the pointer in the field.

When you press *Enter* or *Tab,* or you click another field, the typed value is entered.

The entry method you use determines what is entered in a field when the field is empty. If you press *Enter* without typing a value in the field, an empty string is entered in the field. If you press *Tab* or click another field without typing a value in the field, no value is entered in the field.

The length of the text you type is limited to the size of the text/numeric field (except for scrolling fields). If the field has space for more than one line, your typing wraps into multiple lines as necessary, breaking between words when possible. You

can type up to 4,096 characters for a text value and up to 18 digits for a numeric value.

To show that a number is negative, you can precede it with a minus (–) sign or enclose it in parentheses. Commas used to separate thousands in numbers and currency signs are ignored.

ObjectVision always analyzes the value a user enters to determine whether it is a valid number, date, or time. If so, it converts the value into its internal storage format.

Every text/numeric field has a format and an alignment type associated with it. You can also display Numeric values in a variety of format types:

- **Time:** 03:15PM; 03:15:45 PM; 15:15; 15:15:45; 8/1/90 15:15
- **Date:** 8/1/90; August 1, 1990; 1-Aug-90; 1-Aug; Aug-90
- **Currency:** commas, periods, and currency signs
- **Picture:** see Table 5.1 on page 90 for information about formatting picture fields with match characters.

All numeric values, including dates and times, are converted into an internal format and displayed according to the field's format, not according to the way the data was entered. Therefore, the appearance of the displayed data might be different from the appearance of the data as it is typed by the user.

List fields

In a list field, you select one value from a list of options. List fields reduce typing errors and prevent invalid responses. When you select this type of field, an options list appears within or near the field. After you select an option from the list and press *Enter*, that value displays in the field.

If the field has a previously selected value, that value will be highlighted when you select the field again. You can select a different option from the list by moving the highlight. You can type the first letter of an option's name or press the ↑ or ↓ keys to move the highlight. To enter the selected option as the field value, press *Enter* or *Tab*.

When the number of options exceeds the size of the list box, the list includes a scroll bar on its right side. You can scroll through the options using a mouse or the keyboard. With the mouse, you can click the scroll arrows or slide the scroll box to see more

options. Using the keyboard, you can press *Home* to select the first option in the list and *End* to select the last option in the list.

Database lists are like other list fields, except that they display values read from a database.

Check box fields

Check box fields are like list fields in that you select one option from a set of listed options. However, all the choices display continuously next to check boxes that indicate the selected option.

If the field has a previously selected value, that value is selected (and checked) first. You can select a different option by moving to it and pressing *Spacebar*.

Pressing *Spacebar* changes the state of the check box that is currently highlighted. For example, if the selected check box in an option name is checked, you can press *Spacebar* to uncheck it.

True/False fields

True/false fields are like check box fields with only one option. You check the box to indicate that the field value is true and leave it unchecked (or uncheck it) to indicate that the field value is false.

If the field has a true value, its check box is checked. You can select a false value by pressing *Spacebar* to clear the check box. Pressing *Spacebar* also changes the state of the checkbox from unchecked to checked. You can also click to check and uncheck true/false check box fields.

Scrolling fields

Scrolling fields are the only field type that can contain a value longer than the length of the field. The maximum number of characters a scrolling field can contain is 4,096.

All standard text editing and scrolling techniques are available to users in a scrolling field.

Button fields

Button fields are similar to true/false fields that have a default false value. A button field has a true value only when you choose the button. It returns to a false value after you choose the button.

You activate a selected button field by pressing *Spacebar*. When you press *Enter* or *Tab*, you select the next field without activating the button field. You can also activate a button field by clicking it.

Picture fields

Picture fields are similar to Paradox's PAL pictures, which are used to format user input. Pictures define a pattern that controls values users type into a field during data entry.

Constant values within the picture are displayed in the field as a template for the user to type into. ObjectVision skips over constant characters and beeps when the user attempts to type an invalid character.

For a list of characters ObjectVision uses in pictures and an example of their use, see Table 5.1 on page 90.

Cutting, copying, and pasting data

Besides typing into selected text/numeric fields, you can cut or copy data you want to paste into those fields. Data you cut or copy to the Windows Clipboard remains there until you change it, clear it, or exit Windows. The *Clipboard* is a temporary storage location for data you want to paste into fields or into other applications.

Remember, the Clipboard holds only one piece of information at a time. Each time you cut or copy another piece of information to the Clipboard, the previous item is permanently removed. Pasting information inserts a copy from the Clipboard at the insertion point without removing the contents of the Clipboard.

The paragraphs that follow describe the ObjectVision Edit | Copy, Edit | Cut and Edit | Paste commands.

You can remove selected text and either transfer it to the Clipboard or erase it completely.

Cutting text

The Edit | Cut command deletes the selected text from its field and transfers it to the Clipboard. You can choose this command to remove text from a field and paste it into another field or

application. The Clipboard is also cleared when you choose this command.

Shift+Del is a shortcut for choosing Edit I Cut.

If you want to cut selected text without transferring it to the Clipboard, press *Del*. The current contents of the Clipboard are unaffected.

Copying text

Ctrl+Ins is a shortcut for choosing the Edit I Copy command.

The Edit I Copy command transfers a copy of the selected text to the Clipboard without deleting the selected text. You can choose this command to copy text to paste into another field or another application. You can remove this copy from the Clipboard and restore its previous contents immediately after copying by choosing Edit I Undo.

Ctrl+Ins is a shortcut for choosing the Edit I Copy command.

Pasting text

The Edit I Paste command inserts a copy of the Clipboard contents beginning at the pointer location. You can choose this command to paste text from another field or another application. You can remove the inserted text immediately after pasting by choosing Edit I Undo.

Press Shift+Ins as a shortcut to choosing Edit I Paste.

When you paste data from the Clipboard, the contents of the Clipboard remain unchanged so you can paste the same information into as many places as you want.

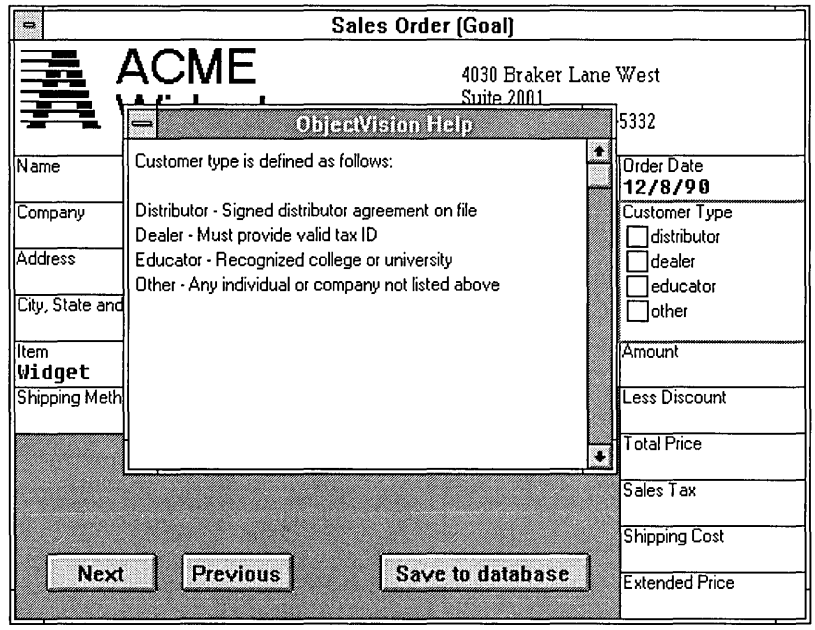
For more information about the Clipboard, refer to your Windows documentation.

Getting help

Online help is always available in ObjectVision, whether you are creating, revising, or filling in an application. You get help by choosing a Help command from the Help menu or by pressing *F1*.

Pressing *F1* displays help for a specific field if the application designer created help for that field *and* that field is selected when a user is in form-completion mode. For example, in Figure 3.3 the selected field is Customer Type and the help window shown appears after *F1* is pressed.

Figure 3.3
The Help window for Order's Customer Type field



After a user finishes reading field-specific help, clicking anywhere outside the help window or pressing *Esc* closes the help window. The help window can also be closed by choosing the Close command from its Control menu.

The field-specific help window can also be moved or resized using the mouse or the corresponding Control menu commands.

A different help window appears when a user presses *F1* and they aren't in form-completion mode or the currently selected field in form-completion mode has no help created by the application designer.

Then, the help that appears is ObjectVision Windows Help, which is a completely different help system from help for a specific field. You can also choose a command from the Help menu, on the right end of the menu bar, to open this Help window.

If a tool is the current window, or a command is highlighted and not yet chosen, pressing *F1* displays help for that command or tool. When neither of these is highlighted, the Help Index appears.

You get help for a tool when either the Form, Tree, or Stack Tool is open. To get help for a command, follow these steps:

1. Select a menu name. You can press *Alt* and use the arrow keys to select a menu, or press *Alt* and type the underlined letter in the menu name.
2. Use the ↑ or ↓ keys to highlight a command name (but don't choose the command by pressing *Enter*). You have to use the arrow keys here instead of the mouse.
3. Press *F1*. Help information appears for the selected command.

The ObjectVision Help window is like any other application window. You can restore, move, resize, minimize, maximize, or close the Help window using the mouse or its Control menu commands.

When a topic contains more help information than fits in the Help window, press *PgDn* or *PgUp* to scroll the text.

If you prefer, you can also click the scroll bar or the scroll arrows.

Many help topics, such as the main index, contain references to related help topics. The titles of related help topics are underlined within the current help text. To display any of these underlined topics, select the underlined text using arrow keys or *Tab*, and then press *Enter*. You can also move to a related topic by clicking an underlined topic name.

Definitions for glossary words appear while you hold down the mouse button. Glossary words are shown with a dotted underline.

The ObjectVision Help window has a menu bar and an icon bar. You can choose menu commands to keep help information handy. For example, File | Print Topic prints the displayed information on your printer, and Bookmark | Define marks a topic so you can easily return to it.

The icons help you locate the help information you want. If the icon is unavailable, it is dimmed. Choose available icons just as you do other Windows buttons.

- **Index**: displays the main index of topics. You can select an underlined topic you want to read.
- **Back**: displays the previous help topic, if any. Use this command to return to last help topic you displayed.

- **Browse <<**: moves to the previous screen in a topic sequence.
- **Browse >>**: moves to the next screen in a topic sequence.
- **Search**: displays the Search dialog box and a list of keywords you can use to find help topics.

For help on using ObjectVision Help, press *F1* or choose Help | Using Help within any ObjectVision Help window to get the Windows Index to Using Help. When you finish, choose the Index button to return to ObjectVision Help.

Clearing forms

When you use an ObjectVision application, you may need to clear a form or a form set (remove all field values) so that you can enter new data into a blank form. Two ObjectVision commands clear forms:

- The Edit | Clear All command clears all the forms in the application.
- The Form | Clear command clears only the currently selected form.

The Edit | Clear All command also removes data from all fields on Scratchpad forms used with the form or form set being cleared.

Neither Edit | Clear All nor Form | Clear clears calculated field values or field values obtained through external links unless a dependent field value is cleared.

You can also use the Form | Clear command for "what if" comparisons. For example, suppose you have completed the *W4Form* sample application and you want to see how changing the values in the Deductions and Adjustments Worksheet form affects the results. You can select the Deductions and Adjustments Worksheet form, clear it with the Form | Clear command, enter new values, and see the changes in the Withholding Certificate form.

Clearing forms with external links

Fields linked to external files are automatically cleared when an index value is changed and there is no external data that matches the new index value.

A Clear button is often included on forms that link to external data, so all of the values associated with the link can be cleared from the form.

Moving forms

You can rearrange the position of forms within the ObjectVision application window at any time. A form first displays in the center of the application window. If the application window is reduced in size and a portion of a form is no longer visible, the form automatically repositions to occupy the center of the application window.

You can move a form from its default location by choosing the Move command from the form's Control menu. You can open the Control menu by pressing *Alt+-* (hyphen) or by dragging the form's title bar with the mouse. Once scroll bars appear around the form, you are unable to move the form. You can position a form so that part of it is outside the workspace, but ObjectVision is unable to select or display fields outside the application window workspace.

Resizing Scratchpad forms

You can't resize an application form while you are completing the form. You must use the Form Tool to change a form's size (as described on page 79).

However, you can use the mouse to change the size of the Scratchpad form by dragging the form's border. Once you have changed the size or location of the Scratchpad form, it retains its new size and location throughout the application.

Scrolling large forms

If a form is wider or longer than the application window, scroll bars are added to the form. Scroll bars are also added when you resize application windows so that a form no longer fits. Scrollable forms are positioned to use as much of the application window as possible. You cannot move a scrollable form. How-

ever, you can scroll the form vertically or horizontally to view other portions of the form.

Forms automatically scroll as necessary to keep the selected field in view.

- To scroll vertically, use the vertical scroll bar. You can click the ↑ or ↓ scroll arrow or drag the scroll box.
- To scroll horizontally, use the horizontal scroll bar. You can click the ← or → scroll arrow or drag the scroll box.

When you scroll a form, the highlight remains visible in the application window. A different field is selected when any portion of a selected field is scrolled outside the application's window. If no field is completely visible, then no field is selected. When no field is selected, you must scroll the form to select a field before you can enter any data.

You can also use the keyboard to scroll a form:

- *PgDn* scrolls a form downward
- *PgUp* scrolls a form upward
- *Ctrl+PgDn* scrolls a form to the right
- *Ctrl+PgUp* scrolls a form to the left

Printing forms

You can print forms to document the current information contained in an application.

- To print only the active form, choose the Print Form command from the File menu.
- To print all the forms in the application, choose the Print All command from the File menu. Each form in the application prints on a separate page.

Before printing forms, you can use the Control Panel application supplied with Windows to select a different printer or to change the attributes of the selected printer. ObjectVision uses the Spooler application for printing, if it is available, to speed up printing. See your Windows documentation for more information on controlling printers in the Control Panel, and enabling and disabling the Spooler.

ObjectVision uses Windows font conventions for both screen display and printing. For best printed results, make sure the appropriate printer fonts are installed for Windows. Because of differences among printers, the exact size and appearance of printed forms depend on the printer you use.

Viewing decision trees

A calculated field has a decision tree associated with it. The decision tree graphically represents the logic used to calculate the field's value. While completing a form, you can examine its decision trees to get a better understanding of how the values are calculated. However, the application designer can restrict the display of any decision tree.

When you want to view the decision tree for a field, first select the field and then choose either Field | Show Tree or Tools | Tree. Field | Show Tree is only available in the form-completion mode. As a shortcut to Field | Show Tree, a user can double-click a field while in form-completion mode.

Tools | Tree can be opened from anywhere in ObjectVision, because the Tools menu is available from every ObjectVision window. As a shortcut to opening Tools | Tree, you can double-click the field if you're in form-edit mode.

Because a ObjectVision Runtime user doesn't have a Tools menu, Field | Show Tree is the only way they can view a decision tree for a field. However, if you choose Properties | Protection, you can check No Tree Display in the Field Protection dialog box to keep users from viewing a field's decision tree.

The decision tree display starts as less than full screen. You can choose the Maximize button or Control | Size from the Control menu to change the size of the window. The tree shown in Figure 3.4 occupies the entire application window.

The name of the field calculated by the decision tree shows in the Title bar of the Field | Show Tree and Tool Tree window (Credit Terms in Figure 3.4). If the values of decision tree branches are available, the current path in the decision tree is marked with a heavy line in the Field | Show Tree window. In Figure 3.4, the selected path within the tree leads from Credit History to Due in 30 days.

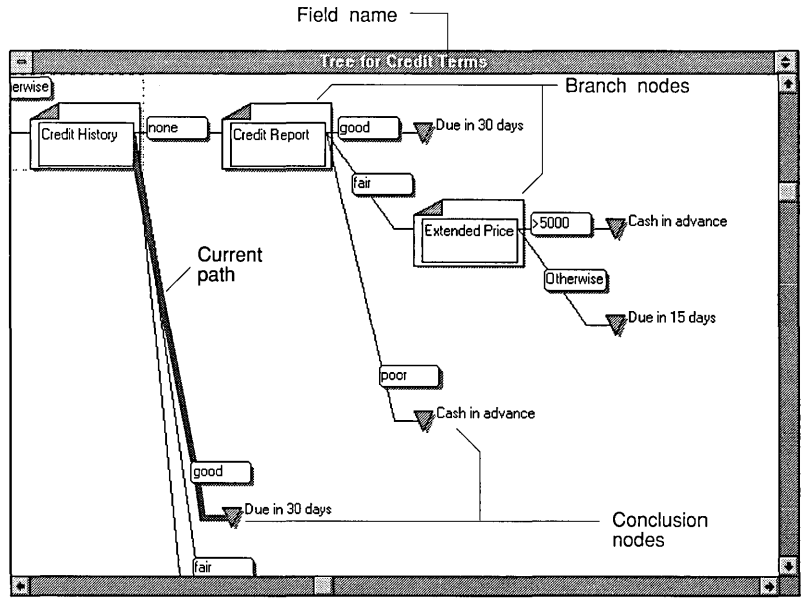
The major parts of a decision tree are called *nodes*. Nodes are connected with path lines. There are two types of nodes:

- Branch nodes identify fields whose values are used in a calculation.
- Conclusion nodes calculate the value of the field associated with the decision tree.

Branch nodes lead to other nodes. Conclusion nodes end a tree segment.

In Figure 3.4, for example, Extended Price is a branch node that gets the price of some merchandise. Cash in advance is a conclusion node that calculates a value that depends on the value of the Extended Price node (if the value of Extended Price is greater than $5000, the value of the Credit Terms field is Cash in advance).

Figure 3.4
Decision tree for Credit Terms



A branch node appears as a stylized form containing the name of its field. A conclusion node displays as an inverted triangle with the conclusion expression to its right.

Each node except the tree root has a *condition* associated with it. The condition is the value that determines whether or not a branch node is evaluated or a conclusion node provides a result. The condition of the preceding branch field leading to a node displays on the path leading to the node.

To provide additional information about decision tree logic, ObjectVision displays a small decision tree icon above the field name of each branch based on a calculated field value. These branches represent a third dimension in the decision logic. ObjectVision must use the logic within the branch field's decision tree to determine its value before it can evaluate the current decision tree.

Because of the limited display area within the application window, sometimes the whole decision tree can't fit within the window. The display is a compromise between the overall structure of the decision tree and the detail associated with each branch.

You can control this trade-off between structure and detail by adjusting the size of the display. You can choose between a high-level view of the decision tree that shows little detail but more of the tree structure or a low-level view that shows more detail but very little structure. This change in size is like zooming out for more structure and zooming in for more detail.

The menu bar changes while a decision tree displays so you can control the display. In Field | Show Tree, these commands are available:

- **Close:** Close the Field | Show Tree window and return to form-completion mode. Choose this command when you finish viewing the decision tree.
- **Expand:** Expand the size of each tree node for additional detail. Choose this command to see field names and expressions for a smaller number of nodes. The window scrolls if necessary to keep the selected tree node visible.
- **Reduce:** Reduce the size of each tree node to display additional structure. Choose this command to see more tree nodes in less detail for each node.

The Tree Tool menu bar includes Edit commands for copying, cutting, and pasting decision tree objects; Objects and Properties menus for creating and editing objects, conditions, and conclusions. The Tree and View menus include these commands:

- **Tree|Select:** Displays another decision tree to be edited.
- **Tree|Print:** Prints the current decision tree in the displayed scale.
- **Tree|Print All:** Prints all the decision trees in the current ObjectVision application using the displayed scale.

- **Tree|Close Tool:** Closes the Tree Tool and returns to either form-completion or form-editing mode (depending on where you were working when you opened the Tree Tool).
- **View|Expand:** Expands the size of each tree node for additional detail. Choose this command to see field names and expressions for a smaller number of nodes. The window scrolls if necessary to keep the selected tree node visible.
- **View|Reduce:** Reduces the size of each tree node to display additional structure. Choose this command to see more tree nodes in less detail for each node.

You can change the currently selected decision tree node by clicking a node with the mouse. If you press an arrow key to select a node outside the current display, the window automatically scrolls to keep the selected node visible. You can double-click any calculated branch node to view the decision tree for that field.

Scrolling decision trees

ObjectVision automatically adds scroll bars to the application window whenever the decision tree is larger than can fit in the application window. You can scroll the window vertically or horizontally to view other portions of the decision tree at the current resolution:

- To scroll up or down using a mouse, click the vertical scroll bar. With the keyboard, press *PgUp* or *PgDn*.
- To scroll to the left or right using a mouse, click the horizontal scroll bar. With the keyboard, press *Ctrl+PgUp* or *Ctrl+PgDn*.

Changing system colors

ObjectVision uses certain colors to enhance the display of forms and decision trees. You can change the colors of display elements with the Control Panel. The paragraphs that follow describe the relationship between form elements and Windows screen element colors.

Forms color

Form backgrounds and text display in the colors you define for Window | Backgrounds and Window | Text. The active form's highlight displays in the color you define for the Active Title Bar.

If the selected field on a form receives user input, the field highlight is a heavy outline in the same color as the Active Title Bar.

If the value of the selected field on a form is calculated by a decision tree or provided by an external link, the highlight is a dashed line inside a heavy outline, and both lines display in the same color as the Active Title Bar.

Decision tree nodes color

Branch nodes display in the color you define for the Menu Bar. Conclusion nodes and the selected path highlight display in the color defined for the Active Title Bar.

Using the Control Panel

You can use the Windows Control Panel to change the default colors, date, time, and currency format Windows uses on your computer. See the Windows documentation that came with Windows for more information.

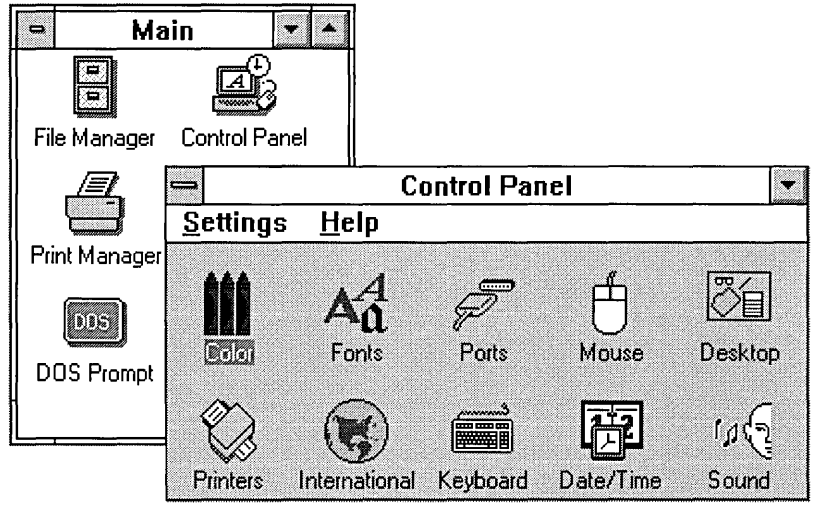
Changing colors

From the Program Manager, open the Control Panel icon. The Control Panel icons appear as shown in Figure 3.5. To change your system colors, open the Color icon. Choose Palette to enlarge the dialog box and display the color choices for Windows.

Any changes you make to your default colors can be saved before closing the Control Panel | Color dialog box. After you finish, choose OK. Windows' colors update to reflect the changes you made.

Note that unusual color combinations, such as a black or green Workspace, can cause portions of the ObjectVision interface to display incorrectly. If your display appears significantly different from the illustrations in this manual, you might want to restore one of the default Windows color schemes.

Figure 3.5
The Control Panel icons

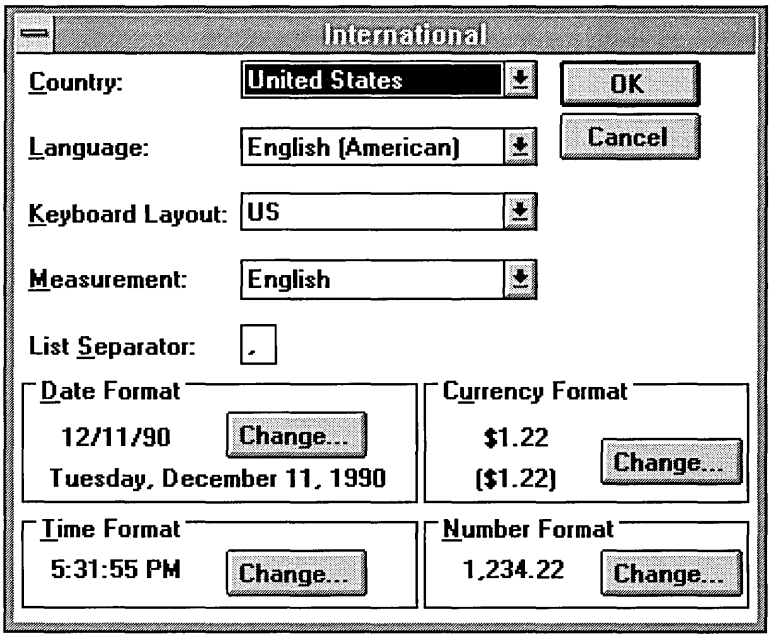


Changing international settings

From the Program Manager, open the Control Panel icon, and then open the International icon. The International window appears, where you can specify settings for the country, language, keyboard layout, measurement, list separators, date format, currency format, time format, and number format.

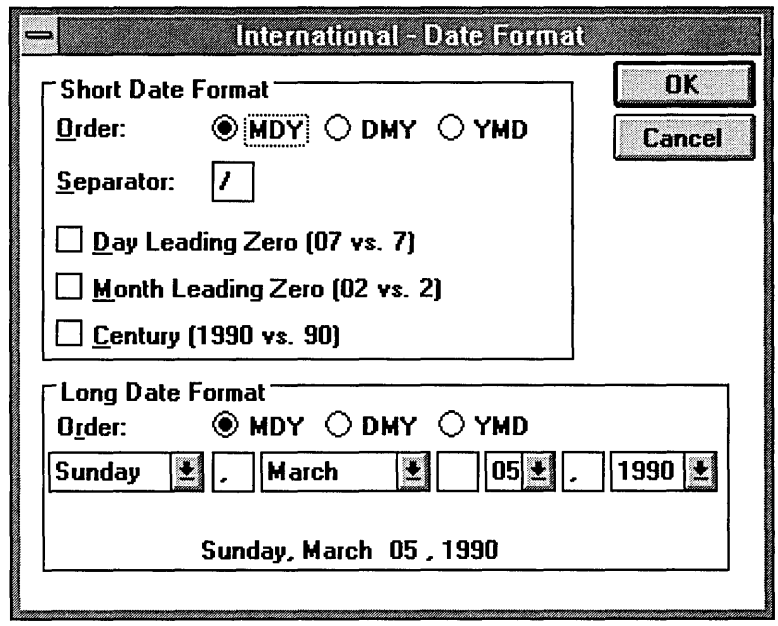
You can specify a particular country's settings by selecting a country name from the Country list. In Figure 3.6, United States is the selected country and its default settings appear for the date, currency, time, and number formats.

Figure 3.6
The International window



You can adjust any of these four default settings by choosing the corresponding Change button, and specifying other options in the window that appears. For example, the International – Date Format window shown in Figure 3.7 appears after you choose Date Format Change.

Figure 3.7
The International – Date Format window



You can also specify the language, keyboard layout, and measurement by selecting the option you want from the list associated with each of these settings.

You preview the changes to your international formats in the International window as you set your options. When you finish, you can choose OK to assign those settings as your new Windows default settings; or, choose Cancel to discard your changes.

If you want your users to use the same settings, you might include step-by-step instructions for them with your run-time application.

C H A P T E R

4

Designing an application

This chapter discusses the form design process you can use to create an ObjectVision application, including

- defining your application's objectives
- creating your prototype
- linking to external data sources
- testing and refining your application
- distributing your application to users
- maintaining and updating your application

Defining your application's objectives

Before you create the first field on your first form, it helps to analyze your application's objectives thoroughly. Major considerations during your analysis phase include

- identifying users of your application
- defining all possible data sets
- listing the expected benefits
- specifying goals for your application
- identifying all possible information sources
- choosing someone to design the forms
- selecting appropriate tools
- estimating design time and costs

It is often useful during the analysis phase to determine your application's required information and final results. You can use

the Form Tool to create a rough prototype of your application's forms.

When your application replaces existing paper forms, you might want to begin by recreating the paper versions onscreen.

At this point, the precise appearance of the forms is not important. You can refine the final format of your forms later. The goal of the analysis phase is to identify what pieces of information are required, which helps define the scope and objectives of the application.

Creating a rough draft of the forms also helps you ensure that all required data is available to your users. By creating the forms early in the process, you can get user suggestions about the content, appearance, and "feel" of your application.

Creating your prototype

Once you create a rough prototype of your forms, you can refine it to demonstrate what your application will do. This refined prototype is valuable for gaining acceptance from users and for demonstrating progress to management.

Your prototype at this phase will lack the full features of your final specification. You need to add enough features in some portion of your prototype to demonstrate how your final application will work on actual data.

For instance, you might limit your prototype to handling only a small number of typical data sets and defer unusual situations until later. You can also limit your prototype by having the user enter data that will eventually be calculated or linked with external sources.

You can refine your prototype using the Tree Tool to add decision trees to fields you defined during the analysis phase. You might want to leave some of the proposed decision trees incomplete or even omit them.

As you add decision trees to your prototype, the overall flow of the application emerges. You can define new fields when you discover the need for them in decision trees. You can also rearrange fields on your forms or add more forms. You can have your users test the flow of your application for ease of use.

Linking your prototype to external data sources

If your application is using external data, your next step is to link your prototype to that external data. You can create links between ObjectVision fields and external data sources, such as databases, so that your prototype operates with "live" data.

Use Tools | Links to add external links to your application. Tools | Links provides a dialog box for defining links to ASCII text files, Paradox and dBASE-compatible databases, and Dynamic Data Exchange (DDE) links to other Windows applications. You can also use link @functions to add external links to decision trees.

Depending on the source of your external data, you may need to perform additional steps to complete the link:

- ASCII files: Make sure both the ASCII files and the applications that read or write those files are available.
- Database information: Make sure that the necessary database and indexes are available to your users' workstations. If the ObjectVision application updates a database, you need to include decision trees or buttons for controlling how users can update.
- Windows application: Establish a DDE link with the other application, referencing your application's fields.

Chapter 10 explains creating, modifying, and deleting external links in detail.

Testing and refining your application

Once you've created your initial prototype, start testing. Make sure that you test the results of all data sets users can enter in the forms. If the prototype is fairly broad in scope and has proven to be accurate, you might consider using it in the actual production environment it is designed for.

Once the prototype performs adequately, design efforts consist of refining and enhancing it. This technique, called *incremental development*, ensures that you always have a working version of your application to test and demonstrate. It also lets you experiment with enhancements but revert to a previous version if you find the results unsatisfactory.

Testing and refining continues until your application gives correct results for all anticipated data sets. In many cases, your application might omit extremely complex situations that occur so infrequently that it is not cost-effective to automate them. When this happens, you need to clearly document these unique conditions.

When your application is nearly complete, you might want to take the time to make it easier to use. For example, you could enhance your application by

- refining the forms to gather and display information more conveniently
- adding help for new users of your application
- modifying the decision tree logic to provide more defaults and shortcuts for experienced application users

Distributing your application to users

Once you've completed and tested your application, you're ready to distribute it to users. Users who will modify the application on their own can run the application in ObjectVision. In other cases, ObjectVision supports distribution with a run-time version that lets users complete forms without giving them the tools to modify the application. See Chapter 11 for more information about ObjectVision Runtime.

ObjectVision Runtime helps control the application design process by requiring users to coordinate requested changes with a form designer. The form designer can approve and implement requested changes, and then distribute updates to all users.

ObjectVision Runtime uses the same application files as Object-Vision. To distribute an application, you simply give users all of the application files, including graphic files, on disk or through a computer network. Each user can then open and use the application with ObjectVision Runtime.

In most cases, you must also provide instructions for using your application. The type of documentation you supply depends on the nature of your application and its intended users.

Maintaining and updating the application

Once you distribute an application to users, the maintenance phase of the design process begins. During this phase you might need to enhance an application periodically to correct errors or add features. The use of a run-time version helps control this process.

Because decision trees provide a high-level graphic representation of the application logic, users can often provide valuable suggestions during the maintenance phase. If the decision tree logic contains errors, users can help identify the exact locations of those errors. If extensions to the decision tree logic are required, users can provide sketches of the enhanced logic flow.

In some cases, you might provide selected users with a licensed copy of ObjectVision so they can make their own changes to the application design. Once a user provides changes, you can test them and then distribute the revised application to other users.

C H A P T E R

5

Using the Form Tool

Forms provide a mechanism for gathering and displaying related pieces of information. ObjectVision provides a specialized tool, called the Form Tool, for creating and modifying application forms. The Form Tool provides a high-level, graphical method for defining forms. It operates much like a drawing package and displays the form as you define it.

This chapter defines the contents of a form and explains how to use the Form Tool. It covers these topics:

- form objects
- creating a new form
- adding new objects to a form
- renaming, resizing, and scrolling forms
- finding forms that contain a specified field
- selecting, moving, and sizing form objects
- editing form objects with the Clipboard
- changing field references
- changing field names and text values
- adding help text to fields
- changing field display formats
- changing the alignment of field values
- changing character fonts
- changing form object borders

- controlling the display of field names
- protecting field values

Form objects

Forms contain fields to display the information entered by the user, calculated by ObjectVision, or provided by an external link. They can also contain other types of information, such as text or pictures, that help identify the form or make it more descriptive.

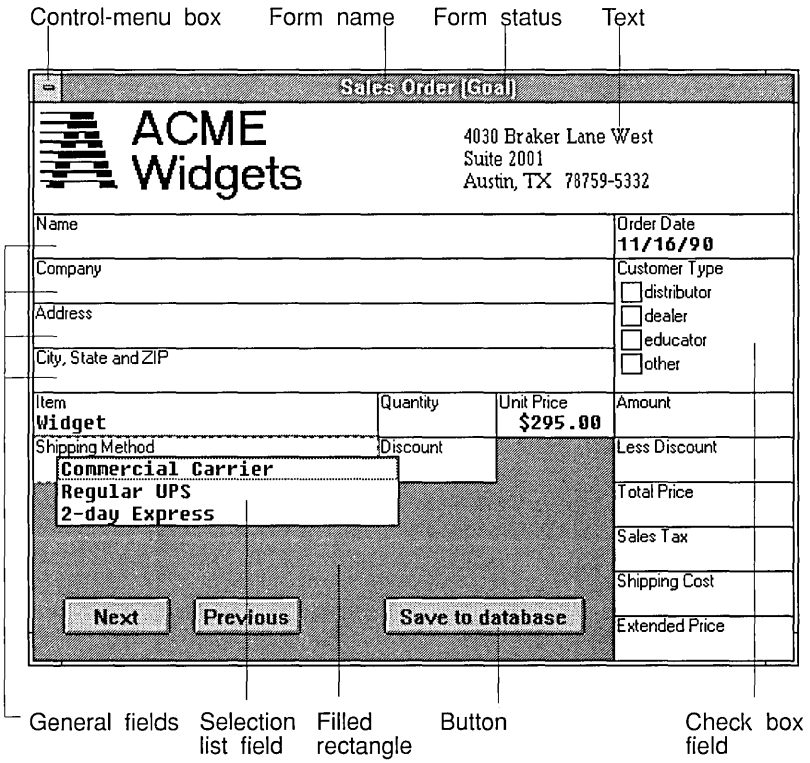
Within ObjectVision, each separate item on a form is called a *form object*. There are six basic types of form objects:

- Fields
- Text
- Filled rectangles
- Rounded rectangles
- Lines
- Graphics

Each form can contain as many objects as you define. You use the Form Tool to define each form object individually and place it on the form. The Form Tool also lets you control various display aspects of each object called *properties*, such as its format and the type of response it requires.

Figure 5.1 shows a form that includes fields, text, filled rectangles, and a graphic.

Figure 5.1
The form window and its objects



In the form shown in Figure 5.1, the fields are the areas containing values. The text objects include the company name, *ACME Widgets*, and its address. The company logo to the left of the name is a graphic. The shaded areas are two adjacent filled rectangles with no borders.

The next sections describe each type of form object and explain how you put objects in a form.

Fields

Fields are the information blanks on forms that contain values supplied during form completion. Each field has a name that can be used to reference its value in decision trees and external links. You can set these properties individually for each field on a form:

- field type
- alignment of values
- label font
- borders

- line width
- protection from modification at runtime
- help for users

Text

Textual information can help identify a form and further describe its contents and use. A text object consists of a constant text string that is always displayed. You can individually format the alignment, label font, and border of each text object on a form.

Filled rectangles

Filled rectangles add shaded areas that can improve the appearance and readability of forms by toning down unused portions of the form. You can select one of fourteen fill patterns from the Properties | Fill Pattern dialog box. You can also format the border and line width of each filled rectangle you put on your form.

Rounded rectangles

Rounded rectangles are similar to filled rectangles, except their corners are rounded. You can use rounded rectangles in the same way as filled rectangles, and assign the same properties to them.

Lines

You can create straight lines on your forms, and assign one of four line thicknesses to them.

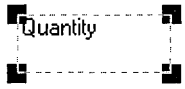
Graphics

A graphic can help identify a form and improve its appearance. A graphic object is a bitmap image or metafile pasted from the Windows Clipboard. You can resize a graphic object and format its border.

Creating and resizing objects

ObjectVision constrains all form objects to a *grid*. The grid size is determined by the default character size of your display.

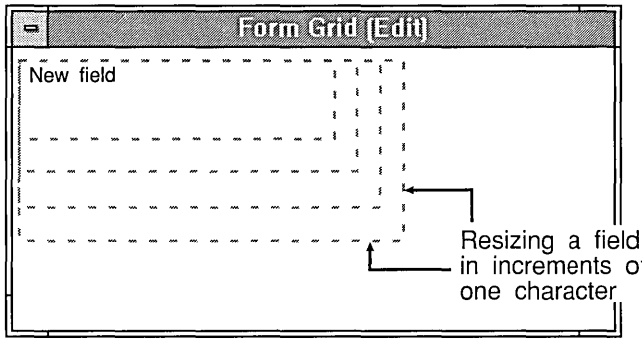
After you choose an Objects menu command (and respond to a dialog box if one appears), your pointer changes to a cross-hair. You position the pointer on one corner or end of the object you want to create.



With the mouse, you drag the object to the size you want. With the keyboard, you press *Enter* to anchor the object position, and then press the arrow keys to define the object's size and shape.

All form objects are created and resized based on an increment of the default character heights and widths. Figure 5.2 illustrates the grid ObjectVision uses when you create or resize objects.

Figure 5.2
Resizing fields



To resize an object, first you select it. With the mouse, position the pointer on one of the black boxes, or *handles*, at the corners or ends of the object, and drag to the new size and shape you want. With the keyboard, first select the object, then press *Shift+* the arrow key corresponding to the new direction you want the object to be resized in.

Creating a new form

You can use the Form Tool to create a new form or to modify an existing form. (You can also create a new form with the Stack Tool; see page 181.) You follow this general procedure to create a new form:

1. While in form completion mode, choose the Form command from the Tools menu.

   The menu bar changes to display menus that are specific to the Form Tool.

2. If you opened an application file before opening the Form Tool, a form is selected for you to edit. To create a new form for the current application, choose the New command from the Form menu. The Form | New | Form Name dialog box appears, where you enter the new form's title.

   If no forms are currently defined when you start the Form Tool (because you have not opened an application file or because you used the File | New command to clear the current application file), the Form Tool immediately prompts you for the title of the new form.

3. Enter a title that is unique within your application. (If you type a form name that already exists, ObjectVision prompts you to type a different name.) You can type a form name of up to 254 characters.

   When you define your form title, remember that form titles too long to fit in your form's title bar are truncated in the form's title bar.

   When you have entered a title, ObjectVision displays a blank form in the default size of 40 characters wide and 10 characters high. This is the form that you'll be adding objects to.

4. Open the Objects menu and choose a command that lets you put an object on the form. For example, choose Objects | Field to put a General text/numeric field on the form; choose Objects | Text to put a text object on the form.

   Now you can either continue to add form objects, or you can work with the object you just inserted, changing its size, for example. As you add field objects, ObjectVision automatically expands the form as necessary.

   As you add fields, remember that when users fill in forms, the field sequence goes from left to right and from top to bottom. ObjectVision determines the order of fields by the position of their bottom right corners.

5. Continue using the Form Tool to create, format, and edit form objects, add help text, or change the size of the form.

   You can close the form at any time, saving your changes, and continue working on it later.

6. When you have finished using the Form Tool, choose the Close Tool command from the Form menu.

   Form | Close Tool closes the Form Tool and returns you to form completion mode. The form you were editing becomes

the new active form. You can save your work by choosing File | Save.

ObjectVision adds new forms to the bottom of the application stack. If you want a form to be in a different position in the stack, you can use the Stack Tool to move it as described on page 181.

Selecting a form

You can have multiple forms open at one time in the Form Tool. You can position forms in the window for easy cutting, copying, and pasting operations between the forms.

When you start the Form Tool, the active form becomes the edit form. You can choose the Select command on the Form menu to display another form you want to edit.

You can also use Form | Select to select an open form that is hidden behind another one in the Form Tool. As a mouse shortcut, you can click any visible portion of a form to select it.

If you are creating a new application, no forms are displayed when you start the Form Tool. The Form Tool opens as a blank window and the Form Name dialog box appears. After you type a form name, a blank form appears in the default size.

Adding new objects to a form

Follow this general procedure to add an object to a form:

1. Select the Objects menu and choose the object you want to add to your form: a field, text, a filled or rounded rectangle, a line, or a graphic.

   ObjectVision prompts you for any necessary information, such as the name of a field. (The sections that follow explain the information needed for each type of form object.)

   The pointer changes to a cross-hair. You move this pointer to indicate one corner or end of the new form object.
2. You can position the pointer at the location you want and then drag the object to define its size and shape.

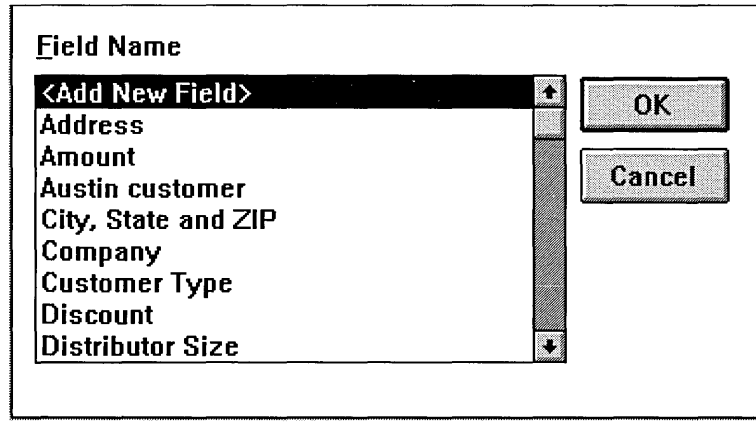
Or, you can press *Enter*, then press the arrow keys to position the pointer, and then press *Enter* again to anchor the pointer. After you anchor the pointer, you can use the arrow keys to define the object's shape and size.

Depending on the type of the new form object, you can also define its appearance with Properties menu commands.

Adding a field

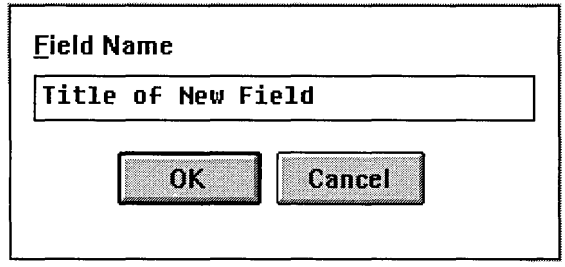
You choose the Field command from the Objects menu to add a new field to your form. When you choose Objects | Field, a dialog box displays an alphabetical list of existing field names. Figure 5.3 shows the sample *Order* application's field names in the Objects | Field dialog box.

Figure 5.3
The Field dialog box



You select <Add New Field>, which appears at the top of the Field | Name list to add a field. After you choose OK, you can enter your new field's name into the Add New Field | Field Name dialog box, as shown in Figure 5.4.

Figure 5.4
Field Name dialog box



Field names must be unique and can contain any text value of up to 254 characters. If the field's name is too large to fit completely within the field's rectangular area, ObjectVision displays as much as will fit. Note, however, that if a field is too small to completely contain the field's name, the field's value won't be displayed, because there's no room left.

Remember, you must make a field large enough to display any of its possible values and a field value can be a maximum of 4,096 characters long. A new field has these default attributes:

- General display format
- Left alignment
- Default character font
- Thin line border outlining the field
- No protection

Later sections in this chapter explain how you can change these attributes.

Adding text

You can use the Objects | Text command to add a new text object to your form. After you choose this command, the Text dialog box lets you type the text you want on your form. Because pressing *Enter* closes the dialog box, you need to press *Ctrl+Enter* when you want to start a new line in a text value. ObjectVision prompts you for the text value to be displayed. You can enter a text string up to a maximum of 4,096 characters. If the text string is too long to fit completely within the text object's boundary, the string is truncated to fit.

A new text object has these default attributes:

- Left alignment
- Default character font
- Thin line border outlining the text

Later sections in this chapter explain how you can change these attributes.

Adding a rectangle object

You can use the Objects | Filled Rectangle or the Objects | Rounded Rectangle command to add a new filled or rounded rectangle

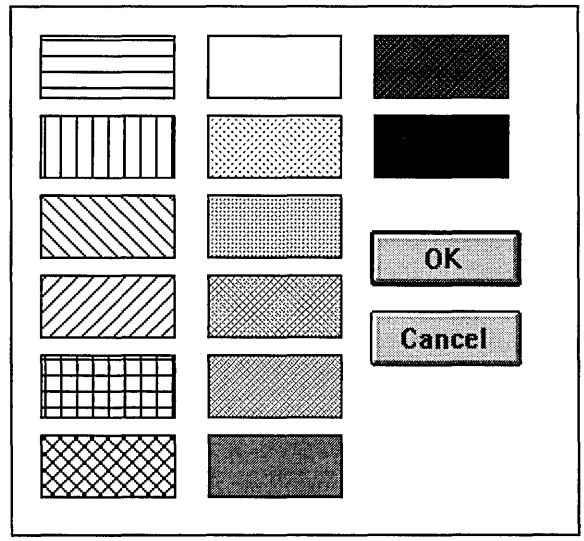
object to your form. You choose Properties | Fill Pattern to display a palette of 14 patterns as illustrated in Figure 5.5. You can select one of the patterns to appear in the selected rectangle.

A new rectangle object has these default attributes:

- Default dot pattern
- Thin line border outlining the rectangle

The section "Changing object borders" later in this chapter explains how to change the border attribute.

Figure 5.5
Properties | Fill Pattern dialog box



Adding lines

Use the Objects | Line command to add a new line to your form. You choose the Properties | Line Width command to select one of four line thicknesses for a selected line. A new line has a default thin line weight.

Adding graphics

Use the Objects | Graphic command to paste an image from the Clipboard to your form.

1. From your Windows graphics application, copy or cut a graphic to the Clipboard.

2. Without exiting Windows, start ObjectVision (if it's not already running), choose the Tools | Form command, and then select the form you want to paste the picture into.
3. When you choose the Graphic command from the Objects menu, the graphic is pasted into your form from the Clipboard.

   ObjectVision supports both Windows metafile and bitmap formats. If both kinds of graphic are on the Clipboard, ObjectVision uses the metafile because it is more device-independent.
4. You can enter a file name for your graphic in the Objects | Graphic dialog box. You can specify any name, but omit the extension; ObjectVision automatically adds the default extension .OVG to your file name.

   The graphic definition is stored in the default directory as a file separate from your application.

When you first paste a graphic, it has a border around it. After you paste a new graphic from the Clipboard, you can resize it like other form objects. If the message `Clipboard does not contain graphic` appears, either the Clipboard is empty or it contains text or numeric information.

Renaming, resizing, moving, and scrolling forms

This section explains how to change a form's title, change the size of a form, move a form, and scroll large forms.

Renaming a form

To change a form's title, select the Properties | Title command within the Stack Tool. Edit the title displayed in the Title dialog box. When you finish, click OK or press *Enter*.

Resizing a form

You can change the size of a form from within the Form Tool. Although ObjectVision automatically adjusts the size of a form to completely contain all of its form objects, you might want to make the form smaller than the default size, or larger to add more whitespace.

To change the size of a form, drag a border of the form with the mouse. With the keyboard, select the form and then open the Control menu by pressing *Alt+-* (hyphen). Choose Control | Size, then press the arrow keys and press *Enter* to finish.

You can't make the form smaller than the size necessary to completely contain all its field objects. If you make the form so large that it can't entirely display within the application window, ObjectVision automatically adds scroll bars to the form.

Moving a form

When you create your form, it is put in a default location. You can move your form with the mouse by dragging its title bar. With the keyboard, you select the form, and choose the Control | Move command; then press the arrow keys to position the form. Press *Enter* when you finish.

Scrolling large forms

If a form is too wide or long to be displayed within the Form Tool, ObjectVision automatically adds scroll bars to the form. Scroll bars are used when a form is larger than the display or when the application window is so reduced in size that a form no longer fits. Scrolling forms are positioned to use as much of the application window as possible and can't be moved.

ObjectVision automatically scrolls forms as necessary to display the selected form object as you move the selection highlight. However, you can scroll the form vertically or horizontally to view other portions of the form:

- To scroll vertically, click the vertical scroll bar.
- To scroll horizontally, click the horizontal scroll bar.

When you scroll a form, ObjectVision selects a different form object if all of the currently selected form object is not visible. If no form object is visible, no object is selected. When no form object is selected, you must scroll the form to select another object before you can perform any edit operations.

Finding forms that contain a specified field

When you are working with the Form Tool and want to see a form that contains a specific field, you can choose the Form | Find command. The Form | Find command displays a list of all the fields you have defined in your application. When you select a field, Form | Find displays the form that contains the field.

If a field appears on multiple forms, you can repeatedly select that field with the Form | Find command to cycle through the forms that contain that field.

For example, if you are working with the Sales Order form in the *Order* sample application and want to see the form that contains the Distributor Size field, you can choose the Find command and select the Distributor Size field. Find displays the Distributor Information form.

Selecting, moving, and resizing form objects

The selected object on the form you are editing is highlighted with a dashed outline. If any form objects are visible, one of them will be shown as the currently selected object. All repositioning, sizing, and formatting commands affect the selected form object.

Selecting form objects

You can select a form object by clicking any portion of that object. When form objects overlap boundaries, continue clicking until the object you want is highlighted.

You can select multiple objects. To select multiple objects with the keyboard, press *Ctrl+Tab* to extend your selection, moving left to right and top to bottom. Press *Ctrl+Shift+Tab* to extend your selection in the opposite direction.

To select multiple objects with the mouse, press *Shift+* click to define a beginning and an ending when selecting a contiguous series of objects. Press *Ctrl+* click to select any set of non-contiguous objects.

Multiple objects can be moved, edited, or assigned properties as a group.

Moving form objects

Once you have selected an object, you can change its position on the form. To move a form object, use the mouse to drag it to its new location. With the keyboard, you select the object and then press the arrow keys to move it.

You can't move a form object past the left or top edge of the form. If you move a form object past the right or bottom edge of the form, ObjectVision automatically increases the size of the form to completely contain the object.

Resizing form objects

To change the size of the selected form object, point at one of the object's handles until the pointer changes shape and then drag the object's corner to its new location.

You can't make a form object smaller than one character wide or one character high. If you increase the size of a form object so that it no longer fits on your form, the size of the form increases until the entire object is displayed.

Using the Form Clipboard

In addition to adding new form objects to the edit form, you can cut, copy, and paste form objects using the Form Clipboard. The Form Clipboard is a temporary storage place for transferring form objects between your forms.

The Form Clipboard is unique to the Form Tool. Objects on the Form Clipboard can't be exchanged with other applications.

When you use the Form Clipboard, remember that it can store only one cut or copied form object at a time. (Multiple objects that are cut or copied as a group are considered a single object.) If you want to insert a cut or copied object into a form, perform the paste operation before cutting or copying any other object.

The following paragraphs describe the editing operations that you can perform with the Form Clipboard.

Cut (Shift Del)

The cut operation deletes the currently selected form object from the current form and transfers it to the Clipboard for later use. Use this operation to remove a form object from the current form. To cut a selected object, choose the Edit | Cut command or press *Shift+Delete*.

Copy (Ctrl Ins)

The copy operation transfers a copy of the currently selected form object to the Clipboard for later use. Use this operation to copy a form object to another form. To invoke the copy operation, choose the Edit | Copy command or press *Ctrl+Insert*.

Paste (Shift Ins)

The paste operation transfers a copy of the form object currently on the Clipboard to the form. Use this operation to insert a form object that was previously copied to the Clipboard from another form. You can position the new form object just as if you were inserting a new object using the Objects menu. The form object will have the same size and attributes as the object that was copied to the Clipboard. To invoke the paste operation, choose the Edit | Paste command or press *Shift+Insert*.

Undo (Alt Backspace)

When you choose the Edit | Undo command, you reverse the previous edit, move, or size operation. You can use this command to undo the effect of a previous operation you made in error. The undo operation remembers only the immediately previous operation. Once you have undone the previous operation, Undo has no more effect until you perform another edit operation. To invoke the undo operation, choose Edit | Undo or press *Alt+Backspace*.

Changing field references

You can change a field object to refer to a different field. To edit the field reference, choose the Field command from the Properties menu. The Properties | Field command is available *only* when a field object is selected.

The Field command is particularly useful in conjunction with the Form Clipboard edit commands. You can use the cut, copy, and paste operations to put copies of a field in other locations within a form or in other forms. These operations transfer all the size and

formatting characteristics associated with the field object. Once you have copied the field and its formatting information, you can then use the Properties | Field command to change the field reference without having to change the location, size, or format of the field object.

You can create a new field using the Properties | Field command:

1. Select a field, then choose the Edit | Copy command.
2. Choose the Edit | Paste command and position the new copy where you want it.
3. Choose the Properties | Field command, select <Add New Field>, and then choose OK. Type a field name into the dialog box, then choose OK.

When you choose Properties | Field, ObjectVision displays an alphabetical list of existing field names and the <Add New Field> option. If you want to create a new field, ObjectVision prompts you for the new field's name. Once you change the field associated with a field object, the form is immediately updated so that you can see the effect of the change.

After changing the field referenced by a field item, you should make sure that the formatting characteristics are still valid for the new field. If the new field assumes different values, you need to change its formatting characteristics accordingly.

Changing field names and text values

You can edit the name of a field or the text associated with a text object. To change the displayed name for a previously created field or to change a text object, choose the Name/Text command from the Properties menu. The Name/Text command is applicable *only* when a field or text object is selected.

When you choose Properties | Name/Text, its dialog box displays the current name of the selected field or the text object label. You can use the mouse or arrow keys to reposition the text pointer. You can also use the standard keyboard edit operations to cut, copy, and paste text.

The field name can be a maximum of 254 characters. Text objects can contain up to 4,096 characters. Press *Enter* when you finish

editing to enter your revisions. Once you enter your changes the updated object is displayed.

Field names within an ObjectVision application must be unique. Names that differ only in capitalization are considered the same and are *not* allowed. If you enter a field name that already exists, ObjectVision will ask you to enter a different name.

When you change the name of a field, ObjectVision automatically changes the name wherever it is used within the application: on forms, in decision trees, or in external links. The name is not automatically changed within any help text associated with the field.

Notice the difference between the Properties | Field command and the Properties | Name/Text command. When you copy a field and use Properties | Name/Text to change its name, the names of both the new copy and the original field change.

Adding help text to fields

To help the user complete your form, you can create field-specific online help for any field.

The user can display field-specific help information during form completion by selecting the field and then choosing the Help command or pressing *F1*. After viewing field-specific help, the user can also choose other help topics from the help system. If the selected field does not have help information associated with it, the help system displays the main topic index when the user requests help.

Help information is associated with the field name. Therefore, when you add help information to a field on any form, the information is available on every form containing that field.

Follow this general procedure to add help information to your fields:

1. Choose Tools | Form to open the Form Tool and begin the form edit mode.
2. Select the field you want to add help information to.

   If you want to add help information to a field that uses the Scratchpad form for prompting, you must first temporarily put the field on an application form and then select it.

3. Select the Help command on the Properties menu. The Properties | Help command is applicable only when a field object is currently selected.

   ObjectVision displays the Help Text dialog box that contains the field's name on the first line.

4. Type your help text into the dialog box, up to a maximum of 4,096 characters.

   You can use the mouse or arrow keys to reposition the text pointer. You can also use the standard edit operations to cut, copy, and paste text. Press *Ctrl+Enter* to begin a new line of text.

5. Press *Enter* when you finish editing and you want to attach your new help information to the selected field.

Adding or changing help text doesn't cause any visible change in the field display.

Changing field display formats

You can specify the appearance of values entered in fields on your form. To control the display format for a field value, choose the Field Type command from the Properties menu. The Properties | Field Type command is applicable only when a field object is currently selected.

The display format of a field object affects only the way the field value is displayed on the form; it does not affect the actual value of the field. In fact, if a field is contained on more than one form, it can have a different display format on each form.

When you choose the Properties | Field Type command, the Field Type dialog box lists the display formats, with the selected field's display format highlighted. When you select a different display format for the field, the form display immediately updates so you can see the effect of the change.

If more than one object is selected, the properties you select are assigned to all objects for which they are appropriate.

This section describes each of the available display formats. Several formats are appropriate for numeric values. All numeric values can contain up to 20 decimal digits (including the decimal point), regardless of how many are displayed on the form.

This section also discusses the Properties | Repeat command, which you can use to apply the last format change you made to the currently highlighted field.

General

The general display format is the default format for newly created fields. It is appropriate for both text and numeric values. The general display format displays numeric values with the minimum number of decimal places to completely specify the number.

Fixed

The fixed display format is appropriate for numeric values only. When you select the fixed display format with the Properties | Field Type command, you can specify how many decimal places to the right of the decimal point will be displayed. The number of decimal places can be any number from 0 to 15. The default is 2. The fixed display format has no effect when the field contains a text value.

Percent

The percent display format is appropriate for numeric values only. The percent display format displays numeric values as a percentage. The value 1.0 is displayed as 100%. When you select the percent format with the Properties | Field Type command, you can specify how many decimal places to the right of the decimal point will be displayed. The number of decimal places can be any number from 0 to 15. The default is 2. The percent display format has no effect when the field contains a text value.

Financial

The financial display format is appropriate for numeric values only. The financial display format separates thousands with commas and displays negative values in parentheses. When you select the financial format with the Properties | Field Type command, you can specify how many decimal places to the right of the decimal point will be displayed. The number of decimal places can be any number from 0 to 15. The default is 2. The financial display format has no effect when the field contains a text value.

Currency

The currency display format is appropriate for numeric values only. The currency display format puts a dollar sign before numeric values, separates thousands with commas, and displays negative values in parentheses. When you select the currency format with the Properties | Field Type command, you can specify how many decimal places to the right of the decimal point will be displayed. The number of decimal places can be any number from 0 to 15. The default is 2. The currency display format has no effect when the field contains a text value.

Date/Time

The date/time display format is appropriate for numeric values only. The date/time display format assumes that the numeric value represents date and time. The integer portion of the number represents the number of days since January 1, 1900. The fractional portion represents time as a fraction of a 24-hour day. The date/time display format has no effect when the field contains a text value.

Once you choose Date | Time, the Date Format dialog box appears. You can select the display format from the following list:

```
8/1/90
August 1,1990
1-Aug-90
1-Aug
Aug-90
03:15 PM
03:15:45 PM
15:15
15:15:45
8/1/90 15:15
```

You can control the format of the first two date formats and the time format by changing the international settings using the International icon in the Windows Control Panel.

Scrolling

The scrolling display format is appropriate only for fields that hold more text than appears on the form. As the user types, the text automatically wraps and scrolls upward.

In addition to the maximum limit of 4,096 characters, a scrolling field value that is linked to an external data file must also observe the limits of the application it is linked to. For example, a value that exceeds 254 characters will be truncated when it is written to a Paradox table.

Selection list

The selection list display format is appropriate only when the value of a field must be one of several possible values. When a selection list field is selected during form completion, the value options the user can select are listed. Once the user selects a value, ObjectVision displays that field value in the general format described above.

The selection list display format has no effect when a list of possible values can't be determined.

The section "Determining possible values," starting on page 92, explains how a selection list field gets a list of possible values.

Check box

The check box display format is appropriate only when the value of a field must be one of several possible values. ObjectVision displays the check box display format as the set of possible choices, with each preceded by a check box. The layout of the check boxes is determined by the number and size of possible choices. If the set of possible values is too large to fit into the field's display area, ObjectVision displays the field in the selection list format.

The user selects a value by checking the box associated with the desired value. The check box display format has no effect when a set of possible values can't be determined.

The section "Determining possible values," starting on page 92, explains how a check box field gets a list of possible values.

True/False

The true/false display format is appropriate for fields that take only Yes or No values. ObjectVision displays the true/false display format as the name of the field preceded by a check box. The box is checked for a Yes value and remains unchecked for a

No value. The check box display format can't be used to display any values except Yes and No.

Button

The button display format is appropriate for fields that take on a momentary Yes value but normally have a No value. ObjectVision displays the button display format as the name of the field surrounded by a button icon. The button takes on a Yes value when activated and then returns to a No value. The button display format is only intended for user input and not for displaying field values.

Your applications can use button fields for link actions with external data sources. For example, in the sample application, *Order*, the Save to Database button updates the Order database using the current form's information.

Note that button fields are *not* printed when the form is printed.

Picture

Choose Properties | Field Type | Picture to assign a picture field. A *picture* is a kind of pattern you specify to control values that users type into a field during data entry. A picture consists of a sequence of literal characters and any of the match characters shown in Table 5.1.

Table 5.1
Match characters in pictures

Picture object	Description
#	Accept only a digit
?	Accept only a letter (lowercase or uppercase)
&	Accept only a letter, convert to uppercase
@	Accept any character
!	Accept any character, convert to uppercase
(see below)†	Any character taken literally

†Any number, letter, or punctuation character not defined as one of the unique match characters (that is, anything you can type that isn't in this table or Table 5.2) is taken literally.

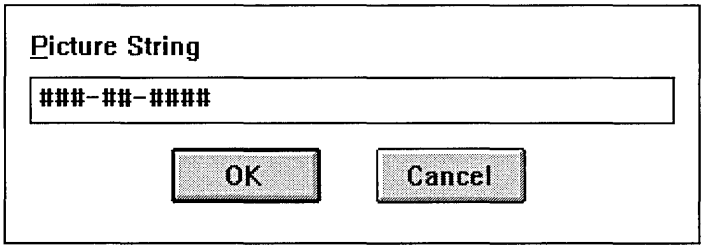
Table 5.2
Special characters in pictures

Picture object	Description
; (see below †)	Take next character literally
*	Reserved for future use
[]	Reserved for future use
{}	Reserved for future use
,	Reserved for future use

†If you need to include a match character or a reserved character as a literal in a picture, precede it with a semicolon. Reserved characters are ignored unless preceded by a semicolon.

For example, suppose you want a user to enter a Social Security number in a field. To ensure the value is typed in the correct format, you might specify a picture in the Picture String dialog box, as shown in Figure 5.6.

Figure 5.6
Picture String dialog box



During data entry, literal characters—like the hyphens in this example—are filled in automatically unless they occur at the beginning of the picture. For example, when the pointer is in a blank field governed by the picture

```
ABC-###
```

the "ABC-" appears automatically and the length of the value is indicated with an underline.

In the Social Security number example, if a user starts by typing a letter, the character does not appear onscreen. If a user tries to leave the field before typing all the digits, ObjectVision displays the message `Field value is incomplete` and leaves the pointer in the field. The only way to avoid filling in a picture is to leave the field blank.

When linking values to data files, remember that a picture such as the Social Security number contains both alphabetic and numeric characters. Your associated database fields must correspond to the values gathered by the picture field. For example, a numeric field

would be inappropriate for the Social Security number, because it has hyphens as well as numbers.

A button field requires a decision tree. You define one or more conclusion nodes in the decision tree to contain system functions to specify the action you want. For example, a conclusion node in the decision tree for Save to Database's button contains the @UPDATE function.

Displaying a field's name

You can display or hide a field's name. When you display it, it appears within the field's margins, using the current text and alignment settings. To turn the display of a field's name on or off, choose the Field Type command from the Properties menu.

The display format of a field object affects only whether its name is displayed on the form; it does not affect the actual value of the field. In some cases you might not want a field's name to be displayed, preferring to have it identified by surrounding text instead. Turning off the field's name is sometimes useful for displaying tabular data or fields with very long or complex identifiers.

When you choose the Properties | Field Type command, its dialog box displays the status of your selected field in the Display Name option. When you select a different status for the name display, the form display is immediately updated so that you can see the effect of the change.

When the field name displays, less space is available to display the field's value. If you remove the field's name from a form, you may be able to make the field smaller. Conversely, if you add the field's name, you may need to make the field larger. Also keep in mind your users, who may need to identify the various fields on your form.

Determining possible values

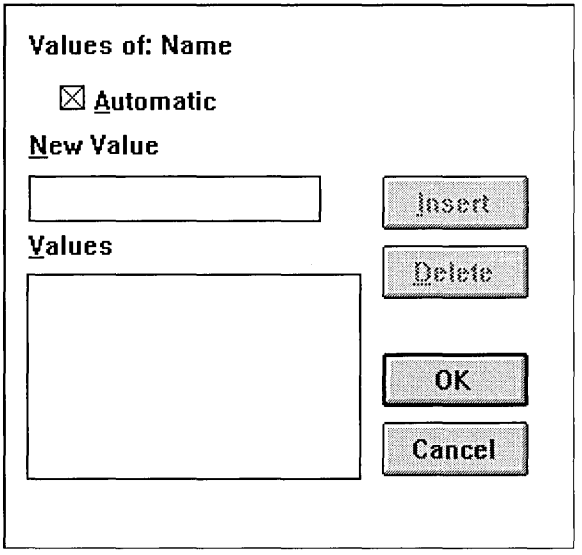
A check box or selection list field gets a list of possible values in one of three ways:

- ObjectVision automatically determines the list from values you define in one or more decision trees. This is the default method.
- ObjectVision reads the value options from a linked data file associated with the ObjectVision field.
- You enter the list of values when you format the field as a check box or selection list field.

When you format a field as a check box or selection list field, the Form Tool displays a dialog box in which you can accept the default, letting ObjectVision determine the list of possible values, or add the values yourself.

Figure 5.7 shows the Values of: dialog box.

Figure 5.7
Values of: dialog box



Note that Automatic is already checked.

The next sections explain the three different methods of providing possible values for selection list and check box fields.

ObjectVision determines the values

When you assign a selection box or check list field type to a field, you don't have to provide a list of possible values (though you can if you prefer). ObjectVision examines the application's decision trees for the field's possible values and adds them. If the field is linked to an external file, the file's values can be used to automatically create the list of value options for the field.

Decision trees can contain field values in two places:

- in constant expressions at the conclusion nodes of the tree
- in constant expressions that are conditions following branch nodes in any tree (where each branch node is the name of the field)

If decision trees contain formula expressions or no decision trees exist that refer to the selected field, ObjectVision is unable to determine a list of possible values.

To illustrate the first case (a set of conclusion nodes with constant expressions), suppose that the check box field Age Group has a decision tree defined for it. The decision tree contains three conclusion nodes with the following values:

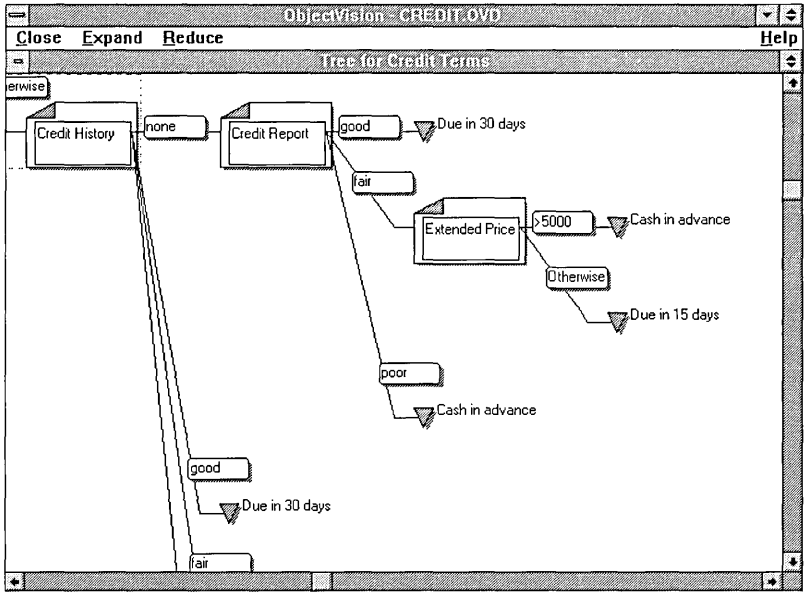
Under 21
21 to 65
66 and above

ObjectVision finds these three values in the decision tree and lists them in the Age Group field, each with a check box.

To illustrate the second case (branch nodes followed by conditions that consist of constant expressions), consider the check box field Credit Report in a credit application.

Credit Report doesn't have a decision tree associated with it, but it's referred to as a branch node in the Credit Terms decision tree. Figure 5.8 shows the Credit Terms decision tree.

Figure 5.8
Decision tree for Credit Terms



Notice that three nodes follow the Credit Report branch node. ObjectVision decides which node to evaluate by looking at the condition (good, fair, or poor) associated with each node. For example, the tree shows that if the value of Credit Report is good, then ObjectVision should evaluate the first succeeding node and calculate a value of Due in 30 days for the Credit Terms field.

The Credit Terms decision tree thus defines three possible values for the Credit Report field: good, fair, and poor. Branch nodes in other decision trees could define more possible values for Credit Report; for example, the decision tree for a field called Authorization Required might have a Credit Report branch node that defines the condition unavailable.

ObjectVision finds all the possible values defined in decision trees and displays them when the field is selected for input.

Adding a list of values

When you're creating a form, you might want to add the list of possible values for a selection list or check box field yourself. Adding the list yourself means that the field gets its list of values right away; otherwise the field won't have a list of possible values until you define the relevant decision tree or trees.

To add a list of possible values yourself, follow these steps:

1. Select the field for which you want to add possible values, start the Form Tool, and choose the Properties | Field Type command.
2. Select Check Boxes or Selection List (whichever is appropriate for the field), then choose OK.
3. The Values of dialog box appears. Automatic is checked, and the Values list is empty.
4. Move the pointer to the New Value field and type the first possible value you want the field to have.
5. Press *Enter*.

   The new value you entered appears in the Values list and the Automatic check box is unchecked, indicating that ObjectVision won't determine the possible values for the field.
6. Repeat steps 3 and 4 to add each possible value that you want the field to have.
7. Choose OK when you finish.

If you later want to add or delete entries, you can choose the Properties | Field Type again for the field, specifying Check Boxes or Selection List. The Values of dialog box appears, displaying the values you have already defined.

Later in the development of your application you might want ObjectVision to determine the field's possible values automatically (perhaps after adding decision trees that define additional or different possible values for the field). Using the Properties | Field Type command with Check Boxes or Selection List, display the Values of dialog box and click the Automatic check box. An *X* will appear in the box, and the entries in the Values field are determined automatically.

Repeating formatting

When you change a field's format with any of the Properties menu commands, you can apply the same formatting to another field using the Properties | Repeat command or its shortcut, *F4*. When you choose the Repeat command from the Properties menu, your last formatting action is applied to the currently selected field.

For example, if you change the display type of one field to currency, you can select another field and then choose the Properties | Repeat command. The field you selected is now formatted for currency display also. You can choose the Properties | Repeat command as many times as you want.

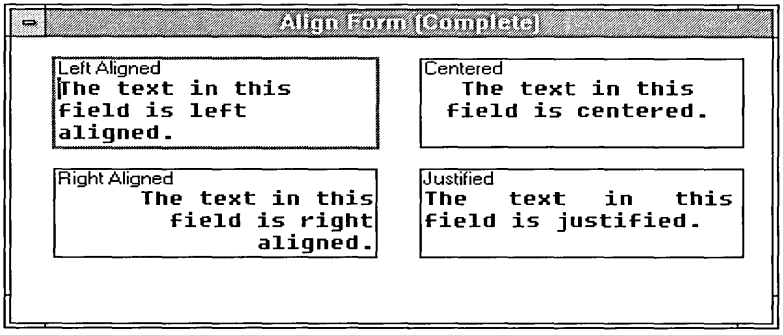
Changing alignment

You can control the alignment of field values and text objects. To specify how the value of a field is aligned on the form, select the Alignment command from the Properties menu. The alignment type also controls the placement of possible responses for check box fields. The Properties | Alignment command is applicable only when a field or text object is currently selected.

For field objects, the alignment type applies only to the field's value. The name of the field, if displayed, is always left aligned at the top of the field's display area.

When you choose the Properties | Alignment command, its dialog box displays a list of four alignment options, Left, Right, Center and Justified. The selected object's current alignment type is highlighted. Once you select a new alignment type for the object, the form display is immediately updated so that you can see the effect of the change. Figure 5.9 shows the different alignment types.

Figure 5.9
Alignment types



The following paragraphs describe each of the available alignment types.

Left

Left-aligned text is the default alignment type for new fields. Left alignment displays field values and text objects starting at the object's left margin.

Right

Right alignment displays field values and text objects with the last character against the object's right margin. Right-aligned text is useful for displaying columns of numbers.

Center

Center alignment displays field values and text objects in the horizontal center of the object's margins.

Justified

Justified alignment displays multiline field values and text objects flush against the object's left and right margins by adding extra spaces between words when necessary. As you type, be sure to let the text wrap. Pressing *Ctrl+Enter* to begin a new line overrides the Justify option. Note that the *last line* of a justified field value or text object is left aligned.

This alignment type recognizes the newline character you create by pressing *Ctrl+Enter*. When you end each line of text with *Ctrl+Enter*, lines are interpreted as single lines that display left-aligned instead of justified.

Changing label fonts

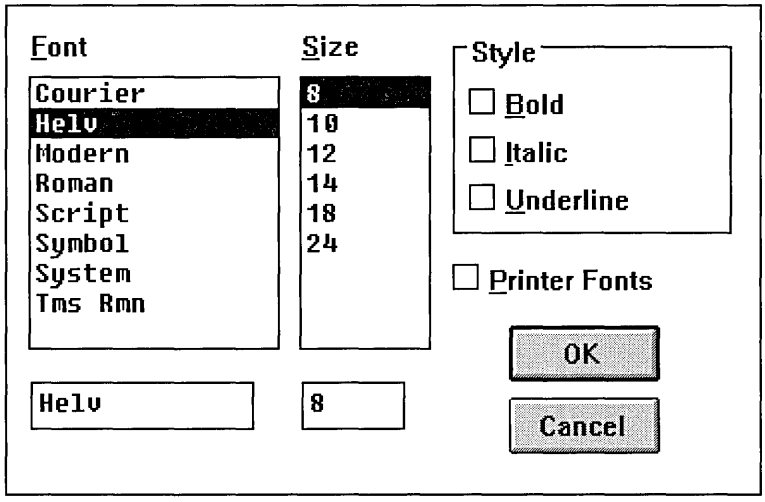
Using the Properties | Label Font dialog box you can control the font for field names and text objects. The default character font for newly created fields and text objects is 8-point Helvetica. You can change the font type, size, and style. The Properties | Label Font command is applicable only to the field or text object currently selected.

For field objects, the font affects only the field's name, *not* its value.

The font also controls the display of value options for check box fields. Field values other than check box fields always display in the system font. When your form is printed, the Courier Font replaces the system font used for field values. For more information, see "Changing screen or printer views" starting on page 100.

When you choose Properties | Label Font, the dialog box shown in Figure 5.10 appears. It lists the available fonts and sizes currently installed on your computer, with the selected object's current character font highlighted. Once you select a different font for an object, the form display immediately updates so you can see the effect of the change.

Figure 5.10
Label Font dialog box



The fonts and font sizes you have installed in your Windows directory display in the Properties | Label Font dialog box. You can select fonts and sizes that are installed for the currently selected printer by checking the Printer Fonts option within the Label Font dialog box.

If you specify a font face or size that is not on the list of available faces and sizes, ObjectVision will select the available font that most closely matches your specification.

See the Windows documentation that came with your copy of Windows for information about installing fonts.

The following paragraphs describe each of the ObjectVision character font attributes.

Fonts

The general style and appearance of characters you type is defined by the font you select. The default font is Helvetica (Helv). ObjectVision lists the available fonts. To select a font,

highlight its name in the list or type the name in the edit field below the list.

Size

Font size is specified in *points*. A point is a unit of measurement used by typographers. A point is roughly equal to 1/72 of an inch. The default font size is 8 points.

ObjectVision lists the available sizes for the currently highlighted font in the size list. To select a point size, highlight the size in the list or type the size in the edit field below the list.

For some fonts, ObjectVision can display large sizes that are not presented in the list by multiplying a smaller font by an integer multiple. For instance, it may be possible to display a 24-point font by doubling the size of a 12-point font. If you would like to use a larger font, experiment to see if ObjectVision can increase the size of the font you are using.

Bold style

The bold attribute determines the thickness or darkness of characters. Check Bold to have characters display thicker than normal (**like this**).

Italic style

The italics attribute determines whether characters are displayed in italic type. Check Italics to have characters display in italics (*like this*).

Underline style

The underline attribute determines whether characters are underlined. Check Underline to have characters displayed with an underline (like this).

Changing screen or printer views

You will notice that your printed form differs from its onscreen display. This is because different fonts are used for displaying and printing. Courier and system fonts are both made up of mono-

spaced characters. All characters within a monospaced font take up the same amount of space horizontally, just as a typewriter's characters do.

However, Courier characters are slightly wider than system characters, so their display is slightly compressed compared to the printed form. For example, some lines in a text object may break at different words when you compare the screen and your printed form. There can also be differences between the displayed and printed versions of proportional fonts like Helvetica or Times Roman.

When you need to see a close approximation of how your printed form will look, you can choose the Printer command from the View menu. You can choose the View | Screen command to return to the default screen display.

The default screen view can be changed for an individual work session, but it can't be changed permanently.

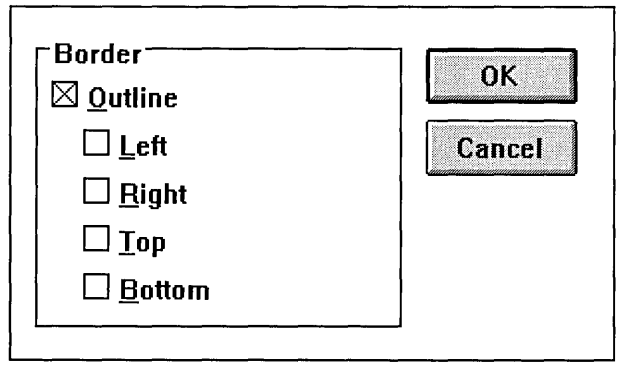
Changing object borders

You can specify whether ObjectVision displays a border around all or part of a form object. To control the display of form object borders, choose the Borders command from the Properties menu. The Borders command is applicable to all types of form objects, except rounded rectangles.

When you choose the Properties | Borders command, ObjectVision displays a list of border styles, with the selected object's current border style checked. When you select a different border style for an object, ObjectVision updates the form display, but you won't see the new border style until you select a different form object.

Figure 5.11 shows the Properties | Borders dialog box.

Figure 5.11
Properties | Borders dialog box



The following paragraphs describe each of the available border styles.

Outline

The outline border style is the default border type for newly created form objects. The outline style displays a line around the entire form object. If you don't want a border, uncheck the Outline option.

Top

The top style displays a horizontal line across the top edge of the form object. You can use this style with any of the other styles except the outline style.

Left

The left style displays a vertical line across the left edge of the form object. You can use this style with any of the other styles except the outline style.

Right

The right style displays a vertical line across the right edge of the form object. You can use this style with any of the other styles except the outline style.

Bottom

The bottom style displays a horizontal line across the bottom edge of the form object. You can use this style with any of the other styles except the outline style.

Protecting field values

You can control whether a user can modify a calculated field value and whether the user can view the decision tree logic that determines the field value. To prevent the user from overriding a calculated value or viewing the decision tree logic, choose the Protection command from the Properties menu. The Properties | Protection command is applicable only when a field object is currently selected.

When you choose the Properties | Protection command, ObjectVision displays two protection options, No Override and No Tree Display, with the selected object's current protection options checked. Selecting a new protection option does not cause a visible change in the field display.

You should only protect calculated fields.

The following paragraphs describe the two protection options.

No Override

The No Override option prevents the user from changing the value of the selected field. This option is initially unchecked for newly created fields.

You can select the No Override option to ensure that the user does not substitute a different value for fields that must always be calculated based on a standard procedure. Since the user can't enter a value for the field, the field can get a value only through calculation. Field override protection does not prevent the user from selecting the field, only from entering a value. The user may still need to select the field to view its decision tree or request that ObjectVision calculate its value.

No Tree Display

The No Tree Display option prevents the user from viewing the decision tree logic associated with the selected field. This option is initially unchecked for newly created fields.

You can select the No Tree Display option when it is preferable that the user not see the logic that determines a field value.

C H A P T E R

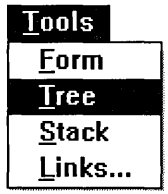
6

Using the Tree Tool

The fields in an ObjectVision form receive values in one of three ways:

- The user enters a value.
- A link to an external source provides a value.
- ObjectVision calculates a value.

For ObjectVision to calculate a field's value, you must define a decision tree for the field. The decision tree tells ObjectVision how to manipulate information that the user enters. For example, to calculate a mileage expense in a travel expenses form, ObjectVision must multiply the number of miles traveled (a value entered by the user) by the number of cents allowed per mile (a number defined within the form logic).



ObjectVision provides a specialized tool, called the Tree Tool, that you use to create and modify decision trees. This chapter explains how to use the Tree Tool and covers these decision tree topics:

- elements of a decision tree
- creating
- editing
- finding
- deleting
- printing

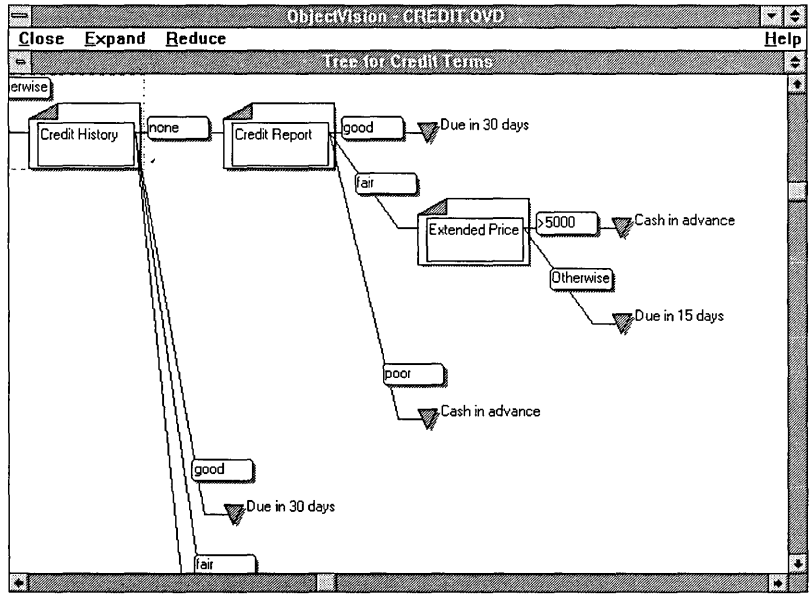
In calculating values, ObjectVision can use expressions defined in decision trees.

Within a form, you must define a decision tree for each field whose value you want ObjectVision to calculate. A decision tree is a graphical expression of decision logic. The nodes (individual segments) of a tree define step-by-step the logic that ObjectVision should follow to calculate a value for the field associated with the tree.

For example, the decision tree defined for the field Mileage expense might include just one node, the expression *+Mileage * 0.2*. This expression tells ObjectVision that to determine the value of the Mileage Expense field, it must multiply by .2 the mileage figure entered by the user. Decision trees can be much more complicated, with many branches and several possible conclusions.

Figure 6.1 shows the decision tree for the Credit Terms field.

Figure 6.1
Decision tree for Credit Terms



This tree shows that the values of the Credit History and Credit Report fields are required to calculate a value for Credit Terms. Depending on that value, ObjectVision may then need to consider the value of the Extended Price field (temporarily leaving the Credit Terms field, if necessary, to determine a value for Extended Price). For example, the tree shows that when the value of Credit Report is poor, the value of Credit Terms is Cash in advance. If the value of Credit Report is fair, ObjectVision must next consider

the value of the Extended Price field. If the value of Extended Price is greater than $5000, ObjectVision can conclude that the value of Credit Terms is Cash in advance. If the value of Extended Price is $5000 or less, the value of Credit Terms is Due in 15 days.

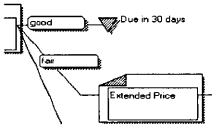
Elements of a decision tree

Decision trees define a decision process as a series of smaller steps. At each step a single field value is evaluated to determine which subsequent branch to take. The decision tree contains a branch for each anticipated value of the field. Each branch can contain additional branches to consider other relevant field values. The conclusion of each decision tree branch is the value to be concluded as the result of the decision process.

Within ObjectVision, every segment of a decision tree is called a node. Each decision tree can contain as many nodes as you define (within the memory limits). You individually define each node and put it in the decision tree using the Tree Tool.

There are two basic types of decision tree nodes: branch nodes and conclusion nodes. A third type of node, an empty node, serves as a placeholder for building decision trees. Other decision tree elements are condition expressions and conclusion expressions. The following paragraphs describe each of these elements.

Branch nodes

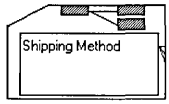


Branches are nodes that evaluate a single field value to determine which subsequent path to follow. Each branch node refers to a single ObjectVision field. The field name displays inside the branch node icon (for example, Quantity).

Simple branch node

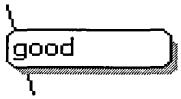
A simple branch node is a node that ObjectVision can evaluate without referring to the decision tree of any other field. The illustration on the left shows the symbol used for simple branch nodes. The entire decision logic for the node is displayed in the Tree Tool when the decision tree for a simple branch node is shown.

Complex branch node



A complex branch node is a mode that ObjectVision can evaluate only by referring to the decision tree of at least one other field. The decision trees of any other fields do not display in the Tree Tool while the decision tree for the currently selected field displays. The illustration on the left shows the symbol used for complex branch nodes. If a complex branch node is double-clicked with the mouse, the associated decision tree is shown in the Tree Tool.

Conditions



All the nodes that follow a branch node, regardless of their type, have a condition associated with them that is compared to the current value of the branch field. The condition for selecting each subsequent node is displayed to the left of the node. In Figure 6.1, for example, the Extended Price branch node follows the Credit Report branch node. The condition associated with Extended Price is fair. The branch structure of the decision tree indicates that when the value of Credit Report is fair, ObjectVision selects the Extended Price node in attempting to calculate a value for the field Credit Terms.

Evaluating condition expressions

ObjectVision evaluates branch conditions starting from the root node (the leftmost node) and moving to the right. When ObjectVision evaluates nodes at the same nesting level (that is, at the same distance from the root node), ObjectVision starts at the top node and continues evaluating condition expressions until one of them evaluates to true. ObjectVision selects the first applicable branch condition even if other conditions below it would also evaluate to true. If none of the branch conditions evaluates to true, ObjectVision can't calculate a value for the field and returns the error value *NA*.

In Figure 6.1, for example, there are two nodes following the Extended Price branch node at the same nesting level. When the value of Credit Report is fair, these two nodes are evaluated from top to bottom. If Extended Price is greater than $5,000, the top condition is met so the value of Credit Terms is Cash in advance. This conclusion is used even though the bottom node's Otherwise condition is always true.

Conclusion nodes

A conclusion node (marked with an inverted triangle placed to the left of an expression in a decision tree) is an end node that supplies a result for the decision process. In Figure 6.1, for example, Due in 30 days is a conclusion node. A tree can contain multiple conclusion nodes. Each conclusion node contains an expression that ObjectVision evaluates when the conclusion is selected (when the path through the decision tree leads to that conclusion node). The value that results from the conclusion node's expression becomes the value of the decision tree and its associated field. (For example, the result of the expression *+Mileage * 0.2* is the value of the Mileage expense field.) The conclusion expression is displayed at the right side of the conclusion node.

Empty nodes

Empty nodes act as placeholders for other nodes while you edit a decision tree. These nodes indicate the absence of a decision tree or the lack of any nodes beneath a branch node. While developing an application, you can leave empty nodes in a tree. However, if ObjectVision encounters an empty node while evaluating a decision tree, an error value results.

Creating a new decision tree

You use the Tree Tool to create a new decision tree (or to modify an existing decision tree). Follow this general procedure to create a decision tree:

1. While you are in form completion or form editing mode, select the field for which you want to create a decision tree. To select a field, you can click any portion of the field or use the Field | Find command.
2. Choose the Tree command from the Tools menu.

   The Tree Tool displays a decision tree that contains a single empty node.

   The Tree Tool menu bar displays menus that are specific to the Tree Tool.

3. Using the Objects menu, add nodes to the tree.
4. When you have finished creating the decision tree, choose the Close Tool command from the Tree menu.

   The Close Tool command closes the Tree Tool and returns you to form completion or form editing mode. ObjectVision evaluates the decision tree you created to determine a new value for the selected field.

The next sections explain how to use the Tree Tool as you create a new decision tree.

Adding decision tree nodes

Once you have selected the Tree Tool, you can add nodes to a decision tree:

1. Select the desired position for the new node.

   You must select the position for the node before adding it because new nodes are always inserted in the decision tree relative to the selected position. (If you do insert the node in the wrong spot, you can cut it and paste it where you want it.)
2. From the Objects menu, choose the command to insert either a branch node or a conclusion node.
3. Provide the information about the new node for which the Objects command prompts you.
4. When you add a branch node, ObjectVision prompts you to define a branch field. When you add a conclusion node, ObjectVision prompts you to define a conclusion expression.

Unless the new node is the root of the decision tree, the Conclusion dialog box appears after you choose either Objects | Branch or Objects | Conclusion. The condition expression you supply refers to the value of the prior branch node that will cause selection of the new node.

Selecting a position for a new node

You can insert nodes either above or below the currently selected node at the same nesting level (the same distance from the root node) as the selected node.

You select a node by clicking it with the mouse. With the keyboard, press the arrow keys to select a node. The new node is inserted immediately below the currently selected node unless you check Insert Above when supplying the new node's condition

expression. If the only node at the current nesting level is an empty node, the new node replaces the empty node.

Adding branch nodes

When you add a new branch node, the New Field Name dialog box prompts you for the name of the field that will supply the branch value. For example, if you were adding the first branch node to the decision tree for Credit Terms shown in Figure 6.1, you would supply the name Credit History.

The Objects | Branch command displays an alphabetical list of existing field names and the Add New Field option. You can either select an existing field name from the list or create a new field.

If you want to create a new field, you are prompted for the new field's name. Field names must be unique and can contain any text value of up to 254 characters.

If you create a new field while using the Tree Tool, the field will not be contained on any form. Unless you later insert the field in a form (using the form tool), ObjectVision uses the scratchpad form to prompt for the field when its value is needed.

When first added, a branch node contains one subsequent empty node. The empty node provides a place holder where you can insert other nodes after the new branch node. The empty node is automatically removed when you insert the first conclusion node.

Adding conclusion nodes

You add add a new conclusion node by entering a conclusion expression in the Objects | Conclusion dialog box.

Editing decision trees

You can use the Tree Tool to modify existing decision trees. The next sections explain how to select a decision tree to edit, how to select and scroll nodes in the tree, and how to edit it with the Tree Clipboard.

Selecting a decision tree

To modify a decision tree, first select it in any of these ways:

- In form completion or form editing mode, select the field whose decision tree you want to edit, and then choose the Tree

command from the Tools menu. The field's decision tree appears so you can edit it.

- If you are already editing a decision tree, you can select a different decision tree with the Select command on the Tree menu. The Tree | Select command closes the current decision tree, with any modifications you have made, and presents a list from which you can select the field whose decision tree you want to edit.
- If you are editing a decision tree, you can double-click a branch node to edit its decision tree. If the branch node already has a decision tree, it displays so that you can edit it. A stylized tree icon appearing inside the node icon indicates the value of the node is determined by another decision tree.

Selecting and scrolling

While you are editing a decision tree, the currently selected node is highlighted with a dashed box. If any decision tree nodes are visible, one of them will be highlighted as the selected node. All editing commands are applied to the currently selected node and any subsequent nodes.

To select a decision tree node with the mouse, click any visible portion of the node.

The size of the display within the Tree Tool is identical to that of the tree viewing mode described in Chapter 3. To see each node in greater detail, you can expand the display with the Expand command from the View menu or press *Ctrl+Home*. The Tree Tool window scrolls if necessary to keep the selected node in view. To see more nodes simultaneously, you can reduce the display size of all nodes by choosing the Reduce command from the View menu or by pressing *Ctrl+End*.

ObjectVision automatically adds scroll bars to the Tree Tool window whenever it is possible to scroll the decision tree. You can also scroll the window vertically or horizontally to view other portions of the decision tree at the current resolution:

- To scroll vertically, one screen at a time, click the vertical scroll bar.
- To scroll vertically, one node at a time, press ↑ or ↓.
- To scroll horizontally, one screen at a time, use the horizontal scroll bar.

■ To scroll horizontally, one nesting level at a time, press ← or →.

If you use the arrow keys to select a node that is not currently displayed, the window automatically scrolls to display the selected node.

Editing with the Tree Clipboard

In addition to adding new nodes to the decision tree, you can cut and paste portions of trees using the Tree Clipboard. The Tree Clipboard is a temporary holding place for portions of decision trees to be rearranged and transferred between fields within ObjectVision. (The Tree Clipboard is distinct from the Windows Clipboard and can't be used to exchange decision trees with other applications, including other copies of ObjectVision.)

When you use the Tree Clipboard, keep in mind that the Clipboard can hold only one cut or copied tree portion at a time. If you want to insert a cut or copied portion into a tree, perform the paste operation before cutting or copying any other portion.

The following paragraphs describe the editing operations that you can perform with the Tree Clipboard.

Undo

Alt ←

The undo operation reverses the effect of the previous edit operation. Use this operation to undo the effect of a previous operation that you made in error. The undo operation remembers only the immediately previous operation. Once you have undone the previous operation, the undo operation has no more effect until you perform another edit operation. To invoke the undo operation, choose the Edit | Undo command. You can also press *Alt+Backspace* as a shortcut.

Cut

Shift Del

The cut operation deletes and transfers the currently selected node and any subsequent nodes to the Clipboard for later use. Use this operation to remove a portion of a decision tree to be placed elsewhere in the decision tree or in a different decision tree. You can also press *Shift+Del* as a shortcut.

Copy

Ctrl Ins

The copy operation transfers a copy of the currently selected node and any subsequent nodes to the Clipboard for later use. Use this operation to copy a portion of a decision tree to be placed elsewhere in the decision tree or in a different decision tree. You can also press *Ctrl+Insert* as a shortcut.

Paste
Shift Ins

The paste operation transfers a copy of the decision tree nodes currently on the Clipboard into the current decision tree. Use this operation to insert a portion of a decision tree that was previously copied or cut onto the Clipboard from elsewhere in the same tree or another decision tree. ObjectVision prompts you to supply a branch condition for the portion of the decision tree that you are inserting. At this point, you can also specify whether the new node goes above or below the currently selected node. To invoke the paste operation, choose the Edit | Paste command. You can also press *Shift+Insert* as a shortcut.

Clear
Del

You can choose the Clear command from the Edit menu to delete a selected node without changing the Clipboard contents. Use this operation to remove a portion of a decision tree so that you can paste the Clipboard contents in its place. You can also press *Del* as a shortcut for this command.

Changing conditions

You can change the condition associated with a node. To change the branch condition, choose the Condition command from the Properties menu (or double click the condition that you want to edit). The Properties | Condition command is applicable for any node that has a condition expression.

When you choose the Properties | Condition, the Condition dialog box appears and displays the current branch condition of the selected node. (See Chapter 6 for information about using the Condition dialog box.) If you enter an invalid expression, ObjectVision prompts you to retype a valid expression. Once you change the branch condition, the tree display is immediately updated so that you can see the effect of the change.

Changing conclusions

You can change the expression that is evaluated for a conclusion node. To edit the conclusion expression for a node, choose the Conclusion command from the Properties menu (or double click the conclusion that you want to edit). The Properties | Conclusion command is applicable only for conclusion nodes.

When you choose Properties | Conclusion, ObjectVision displays a Conclusion dialog box that contains the current conclusion

expression of the selected node. (See Chapter 6 for information about using the Conclusion dialog box.) If you enter an invalid expression, ObjectVision prompts you to retype a valid expression. Once you change the conclusion expression, the tree display is immediately updated so that you can see the effect of the change.

Changing branch field references

You can change a previously created branch node to refer to a different field value. To change the field associated with a branch node, first select the branch node, and then select the Field command from the Properties menu. The Properties | Field command is available for this *only* when a branch node is selected.

The Properties | Field command is particularly useful in conjunction with the Clipboard edit commands. You can use the cut, copy, and paste operations to copy portions of decision tree logic elsewhere in the same decision tree or in another decision tree. These operations transfer all of the subsequent nodes and branch conditions associated with the edited node. Once you have copied the decision tree logic, you can then use the Properties | Field command to change the field referenced by a branch node without changing the overall structure of the logic.

When you choose the Properties | Field command, ObjectVision displays an alphabetical list of existing field names and a new name option. If you want to create a new field, ObjectVision prompts you for the new field's name. Once you change the field associated with a branch node, the tree display is immediately updated so that you can see the effect of the change.

After changing the field referenced by a branch node, you should make sure that any subsequent branch conditions are still valid for the new field. If the new field assumes values different from those of the previous field, you may need to edit the subsequent branch conditions accordingly.

Changing branch node field names

You can change the displayed name for previously created branch nodes. To edit the name of a field associated with a branch node, choose the Name command from the Properties menu. The Properties | Name command is applicable only when a branch node is currently selected.

When you choose the Properties | Name command, ObjectVision displays an edit window that contains the current name of the selected field. You can use the arrow keys and mouse to reposition the pointer within the text. You can also use the standard keyboard edit operations to cut, copy, and paste text within the editor. The name text can contain up to 254 characters. Press *Enter* to terminate the edit and enter the new name into ObjectVision. Once you change the name of an element, the tree display is immediately updated so that you can see the effect of the change.

Field names within an ObjectVision application must be unique. Names that differ only in capitalization are considered the same and are not allowed. If you enter a field name that already exists, ObjectVision displays a warning message and cancels the command.

When you change the name of a field, ObjectVision automatically changes the name wherever it is used within the application on forms, in other decision trees, or in external links. (The name is not automatically changed within any help text associated with the field.)

Finding decision trees containing a specified field

When you are working with the Tree Tool and want to see a tree that contains a specific field, you can choose the Tree | Find command. Tree | Find displays a list of all the fields you have defined in your application. When you select a field, Find displays the tree that contains the field. If the field appears in multiple trees or multiple times in the same tree, you can repeatedly select that field with the Tree | Find command until the additional trees are found.

Deleting decision trees

You can completely remove a decision tree from a field by first selecting the field while in form completion mode, then choosing the Tree command from the Tools menu, and then deleting the entire decision tree structure using Edit | Cut or Edit | Clear.

Printing decision trees

You can print decision trees from within ObjectVision to document the decision process of an application:

- To print only the current decision tree, choose the Print command from the Tree menu.
- To print all the decision trees in the application, choose the Print All command from the Tree menu.

Decision trees are printed on multiple pages if required. Large trees are printed in vertical strips that contain all of the nodes within a band of nesting levels. Each piece of paper contains the same number of nodes horizontally as are displayed within the Tree Tool.

You can control the resolution of the decision tree printouts by adjusting the resolution of the tree display (with the Expand and Reduce commands on the View menu) before printing the tree definitions. You can thus control the trade-off between the amount of detail presented and the number of pages required to hold the entire decision tree.

ObjectVision prints decision trees on the printer that is currently selected for use with Windows. Before printing decision trees, you can use the Control Panel application supplied with Windows to select a different printer or to change the attributes of the selected printer. During printing, ObjectVision uses the Windows Print Manager to speed the printing process. See your Windows documentation for information on using the Print Manager.

C H A P T E R

7

Writing expressions

You use the Tree Tool to create expressions in decision trees. Expressions are a sequence of values and operators that ObjectVision evaluates to determine field values.

You can use any of ObjectVision's data types, @functions, or your application's field names in an expression. You can determine what data type a field has by using the @TYPE function. @Functions are discussed thoroughly in Chapter 8, "@Function commands."

Expressions

ObjectVision expressions are compatible with Quattro Pro's formulas. If you know how to enter Quattro Pro formulas, you already know how to enter ObjectVision expressions.

Expressions can be up to 4,096 characters long. You can include spaces between operators and values, but ObjectVision deletes them automatically. If a conclusion or condition begins with one of these characters

0 1 2 3 4 5 6 7 8 9 . + - (@

it is evaluated as an expression. If a condition or a conclusion does not begin with one of these characters, the entire expression is treated as a *label*. A label is a constant string value that doesn't have to be enclosed in double-quotation marks.

You can use the *label prefix*, a single-quotation mark ('), as the first character of an expression to force that expression to be evaluated as a label. When ObjectVision evaluates a label, all characters on the same line of the expression are interpreted as literal characters.

ObjectVision data types

When an ObjectVision field has a value it is one of four underlying data types: numeric, string, logical, and error.

Numeric values

Numeric values are numbers such as *1048*, *–84*, or *43.23*.

Numeric constants can contain a combination of decimal digits without internal spaces.

ObjectVision does not support an exponential format for numeric values in expressions. Commas should not be used to separate thousands in numbers.

String values

String values are a sequence of characters. A string can contain from 0 to 4,096 characters.

Logical values

Logical values are an extension to Quatro Pro data types and are reserved keywords in ObjectVision. Expressions which are true/false statements concerning values in other fields such as *+Quantity < Credit Allowed* return logical values. ObjectVision checks the fields to see if the statement is true or not. If true, it returns a Yes; otherwise, it returns a No.

These operators return a logical value:

= <> < > <= >=

and are typically used with these @functions:

@AND	@NOT
@IF	@OR

ObjectVision uses a special internal representation for the logical values Yes and No, and their logical equivalents TRUE and FALSE. Logical values are recognized without regard to case.

Error values

ObjectVision returns ERR or NA when an expression error is encountered. You can specify a constant error value as the result of an expression using the @ERR or @NA functions. ObjectVision error values are compatible with the error values produced and recognized by Quattro Pro.

The error values and their meanings are

NA	Unanticipated branch condition
ERR	Invalid value

For more information about ObjectVision error values, see Chapter 8, "@Function commands."

Data conversion

While evaluating an expression, ObjectVision attempts to perform any necessary data conversions. For operations and @functions that require numbers, text values are converted into numbers when possible.

Examples

For example, the result of the expression +"*23*"+*3* is *26*. The text string "23" is converted to the number 23 for the addition operation.

Similarly, the result of the expression *Yes*+*Yes* is *2*. The logical value Yes is converted to 1 for the addition operation.

The result of the expression +"*42*"*&3* is "*423*". The numeric value 3 is converted to a "3" string for the concatenation operation.

An ERR error results when ObjectVision is unable to convert an argument to the necessary type.

Table 7.1 shows the complete set of data conversion rules. The Error data type is not included in the table because any error value in an expression causes the result of the expression to be that error value.

Table 7.1
Automatic data conversion in expressions

	Numeric result	String result	Logical result
Numeric data	No conversion required.	Converts the number to a string form of the number in general format.	Converts 0 to No, all other numbers to Yes.
String data	Converts the string to a number, or ERR. The empty string "" is converted to 0.	No conversion required.	Converts the string to a logical value, or ERR.
Logical data	Converts Yes to 1; converts No to 0.	Converts Yes to the string "Yes"; converts No to the string "No".	No conversion required.

Field names in expressions

You can include field names in an expression to refer to the field's current value. If you type a field name in an expression and the name contains characters that ObjectVision may interpret as part of the expression syntax, you *must* enclose the name within single-quotation marks.

If you use the Paste Field button in the Condition and Conclusion dialog boxes described on page 129 to include a field name in an expression, the quotation marks are automatically added.

Single quotation marks

Single quotation marks are required in four situations:

- Your field name starts with any character other than a letter, such as *'2nd Choice'*.
- The last character of your field name is a blank, such as *'Other . . . '*.
- Your field name contains any character that is not a letter, number, space, period, or underscore. For example, *'City/State'*, *'Self-Employed'*, and *'US Citizen?'*.
- Your field name is one of the reserved keywords *Yes*, *No*, *TRUE*, or *FALSE*.

In addition, you must use two single-quotation marks when a single quote is part of a field name. You must then enclose the entire field name in single quotation marks as well because it contains a character that is not a letter, number, space, period, or underscore. For example, to enter the field name *Spouse's Occupation* in an expression, type `'Spouse''s Occupation'`.

When you enter a field name in an expression, it is unnecessary to match the case of the field name.

If an expression refers to a field name that does not exist, ObjectVision automatically creates a new field by that name. Unless you later add the field to a form using the Form Tool, ObjectVision will prompt for this field using the Scratchpad form.

When a field name is in an expression, the expression is evaluated *only* if the field has a value.

To avoid inadvertently creating new fields, you can use the Paste Field button instead of typing the field name. Pasting the name eliminates typographical errors and makes it easy for you to include a field name in a formula.

Operators and evaluation

Expressions use operators, or mathematical symbols, to express a relationship between two or more values. Any blanks before or after the operator are ignored. The result of an expression depends on the order in which ObjectVision performs the arithmetic operations. ObjectVision assigns each operator a *precedence* and performs the operations in order of precedence.

Table 7.2 lists the ObjectVision expression operators and the precedence assigned to each. Operators with the highest precedence (7) are performed *first*.

Table 7.2
Expression operators

Operator	Operation	Precedence
&	String concatenation	1
=	Equal	2
<>	Not equal	2
<	Less than	2
>	Greater than	2
<=	Less than or equal	2
>=	Greater than or equal	2
+	Addition	3
–	Subtraction	3
*	Multiplication	4
/	Division	4
–	Negative	5
+	Positive	5
^	To the power of (exponential)	6
%	Percent	7

You can override the precedence of operators by including parentheses in your expression.

Using parentheses

To ensure that ObjectVision evaluates an expression in the order you want, use parentheses to enclose the portion you want calculated first. You can nest parentheses inside other parentheses; ObjectVision calculates the innermost set of parentheses first. For example,

```
4 * 2 + 3 = 11
4 * (2 + 3) = 20
(4 * 2) + (3 + 5) * 4 = 40
((4 * 2) + (3 + 5)) * 4 = 64
```

If you don't use parentheses, ObjectVision performs the calculations in the order (precedence) shown in Table 7.2.

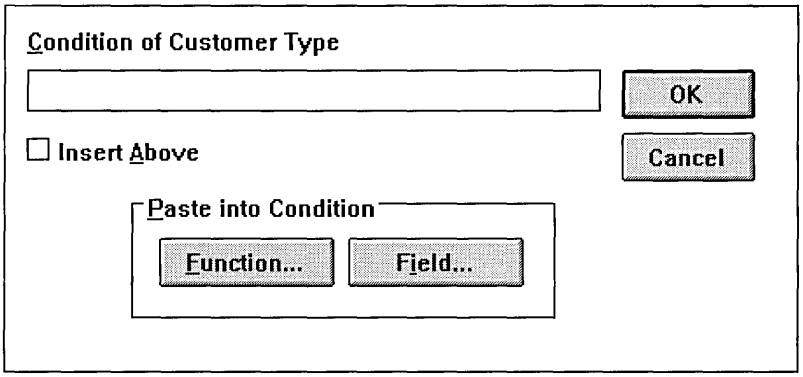
The Condition dialog box

Within decision trees, expressions specify which condition determines which branch node path ObjectVision evaluates. Each node in a decision tree—except the root node—has a condition expression associated with it.

Whenever you insert a new node into a decision tree at a position other than the tree's root, the Condition dialog box appears so you

can supply the condition associated with the new node. Figure 7.1 illustrates the Condition dialog box.

Figure 7.1
Condition dialog box



You can enter an expression of up to 4,096 characters in the Condition text box. If the expression is longer than the text box, your text scrolls horizontally to let you type or edit the entire expression.

You can use the standard editing operations to type and revise your expression.

After you finish and choose OK, ObjectVision checks the expression for validity. If the expression is invalid, a message displays indicating the type of the first detected error. You can then revise the expression to remove the error.

You can stop editing a condition expression at any time by choosing Cancel. When you are defining a new node for a decision tree, choosing Cancel also cancels adding the new node. When you are modifying an existing node's condition, choosing Cancel leaves the previous condition expression unchanged.

Evaluation

1. ObjectVision evaluates the condition expression whenever the branch field's value changes and a subsequent branch must be evaluated. ObjectVision evaluates branch conditions from top to bottom until obtaining a true result. After a true result is obtained from a condition expression, other condition expressions at the same nesting level are ignored.

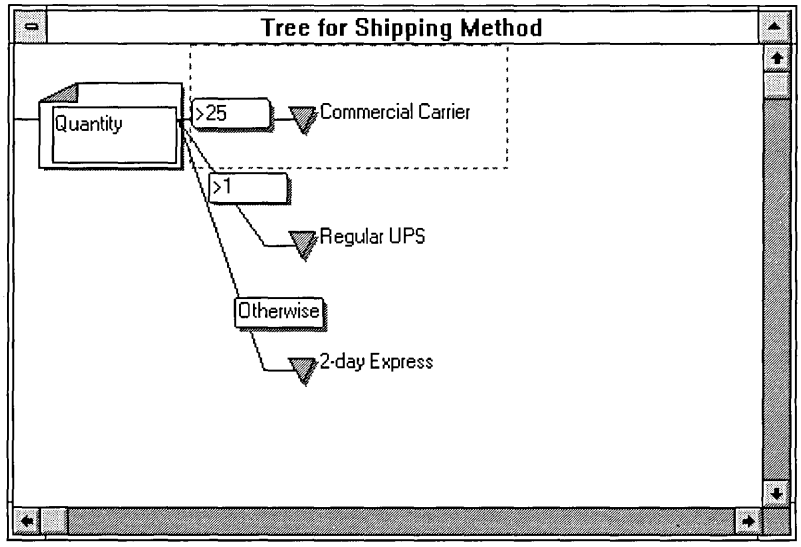
2. The condition expression is evaluated as follows:

   branch-value comparison-operator condition-expression

3. In this context, *branch-value* is the current value of the branch field. Because branch-value is known from the node's location in the decision tree, you omit the name of the branch field. The result of your condition expression is compared to the branch value to determine a true or false result for the condition.
4. The *comparison-operator* is the operator that begins the condition expression. If the condition expression does not begin with a comparison operator, = is assumed.
5. The *condition-expression* portion of the expression is any valid combination of constant values, operators, functions, and field names.
6. If the initial comparison operator is not included in your expression (the = is implied) you can use the single-quotation mark method to force the expression to be evaluated as a string constant.
7. The result of the completed evaluation of a branch condition is always either Yes or No, never one of the error values.
8. Figure 7.2 illustrates the decision tree for Shipping Method in the sample application *Order*. Shipping Method's decision tree includes the following expression after the branch node Quantity:

   >25

Figure 7.2
Decision tree for Shipping Method



Here, *branch-value* is the current value of Quantity. *Comparison-operator* is the greater-than operator, and *condition-expression* is the constant value 25.

Comparison operators

You can use any of the following comparison operators to precede an expression:

= <> < > <= >=

If you don't begin your condition expression with a comparison operator, = is assumed.

ObjectVision always evaluates an expression *before* comparing its result with the branch field value.

Otherwise

ObjectVision uses the Otherwise constant expression to select a branch without regard to the branch field's value. Otherwise is a special constant condition expression that always evaluates to true. You can use Otherwise to handle any situations you have not explicitly defined.

It is usually the last branch condition but its placement is not important; all other branch conditions are checked before the Otherwise branch path is chosen.

Do not use a single-quotation mark before Otherwise when you enter this condition. That represents the string "Otherwise" instead of the special branch condition Otherwise.

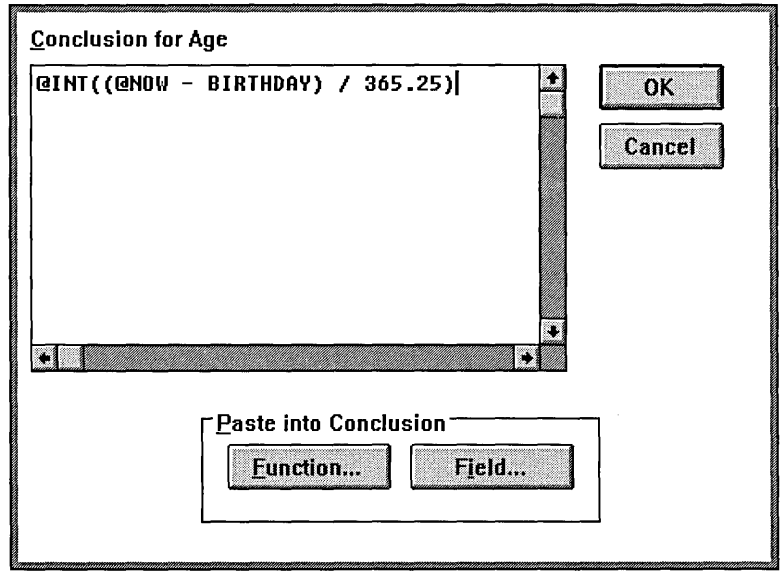
The Conclusion dialog box

Within decision trees, expressions specify the result of each conclusion node.

Conclusions can contain multiple expressions, with each one on a separate line. Multiple expressions allow a conclusion to perform more than one action. When a conclusion node is calculated, its expressions are evaluated in the order you defined. When a conclusion contains multiple lines, the result of the last expression is the new value for the calculated field.

Whenever you insert or modify a conclusion node, you specify the expression associated with that node in the Conclusion dialog box, as illustrated in Figure 7.3.

Figure 7.3
Conclusion dialog box



For example, this sample conclusion expression calculates an age from a birth date using @functions and a Birthdate field.

@INT((@NOW-Birthdate)/365.25)

@NOW returns the current date and time from the user's computer clock as a serial number, and @INT truncates the result by removing the time portion of the number.

You can enter an expression of up to 4,096 characters in the Conclusion text box. If your expression is longer than the Conclusion text box, the text scrolls horizontally or vertically to let you type or edit the entire expression.

You can use editing operations to revise the expression. Because pressing *Enter* chooses OK and closes the dialog box, you must press *Ctrl+Enter* to add another line at the current pointer location.

After you enter an expression in the Conclusion dialog box, the expression's validity is checked. If the expression is an invalid conclusion, a message displays indicating the type of the first detected error. You can revise the expression to remove the error.

You can stop editing a conclusion expression at any time by choosing Cancel. When you are defining a new node for a decision tree, choosing Cancel also cancels adding the new node. When you are modifying an existing node's conclusion, choosing Cancel leaves the previous conclusion expression unchanged.

Evaluation

ObjectVision evaluates a conclusion expression whenever that conclusion node is at the end of the selected path through the decision tree. The result of the expression becomes the current value of the field being calculated.

The expression that you enter in the Conclusion dialog box is evaluated as follows:

calculated-field = *conclusion-expression*

Calculated-field is the field associated with the decision tree. Because the calculated field is associated with the decision tree, you omit the name of the field. The result of the conclusion expression is entered into the calculated field.

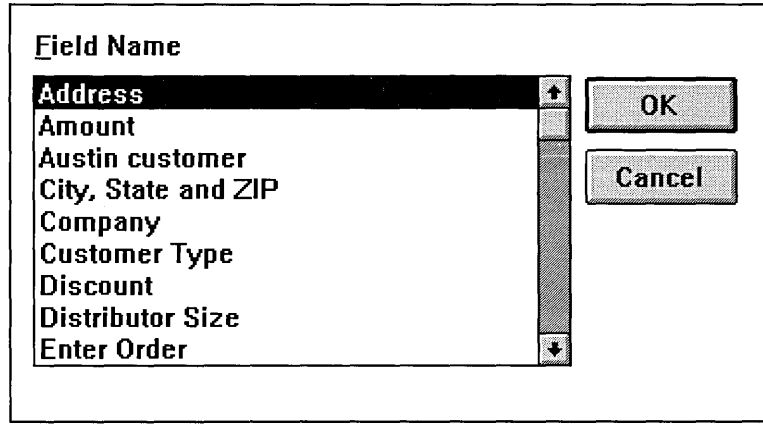
In this case, = represents assignment, *not* the comparison operator.

Pasting field and function names

In both the Condition and Conclusion dialog boxes, you might want to specify field names or functions as part of a formula expression.

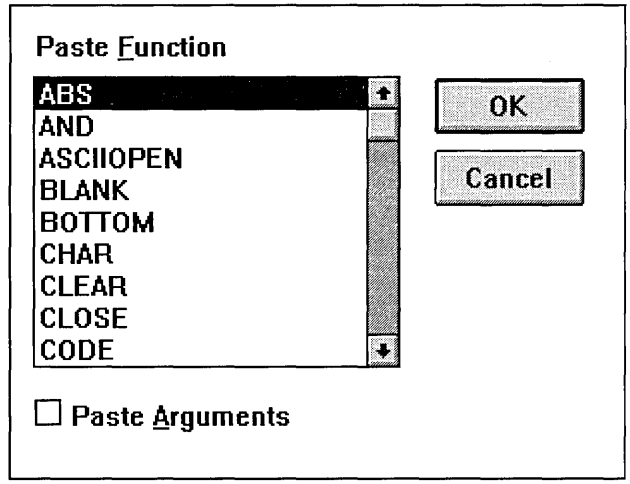
The easy way to specify field names is to choose the Paste Field button in the Condition and Conclusion dialog boxes. The Paste Field | Field Name dialog box lists all fields in your application, as shown in Figure 7.4.

Figure 7.4
Paste Field | Field Name dialog box for Order



Similarly, the easy way to specify @functions is to choose the Paste Function button in the Condition and Conclusion dialog boxes. You can select functions you want to paste into your expression from the list in the Paste Function dialog box shown in Figure 7.5.

Figure 7.5
Paste Function dialog box



Because these dialog boxes list the currently defined fields or ObjectVision functions, the paste function is useful when you are unsure of the precise name of a field or function. Additionally, you might want to paste dummy arguments into your function by checking Paste Arguments. Dummy arguments display the argument list for each @function.

Pasting field names

Follow these steps to paste a field name into a conclusion expression:

1. Select a field you want to add a conclusion expression to.
2. Choose Tools | Tree to start the Tree Tool.
3. Choose Object | Conclusion and type the plus sign (+) into the Conclusion dialog box.

   If Object | Conclusion is unavailable, you selected a field that already has a conclusion. Select another field and try again.
4. Choose Paste Field. The Object | Conclusion | Paste Field | Field Name dialog box displays an alphabetical list of field names.
5. Select the field name you want from the list.
6. Paste the name into the current expression by pressing *Enter* or by choosing OK. As a mouse shortcut, you can double-click the field name you want to select.

   If the selected field name contains one or more characters that are recognized as part of the expression syntax and would

cause syntax errors in the expression, ObjectVision automatically adds single quotation marks around the field's name.

You cannot define a new field name with the Paste Field function. If you omit the plus sign before the field name, the field name is entered as a literal string, beginning with a single quotation mark.

The Field Paste function is recommended over typing the field name. If you type the field name and misspell it, a new field is created with the misspelled name.

When you double-click a node, its appropriate Condition or Conclusion dialog box appears.

Pasting function names

Follow these steps to paste a function name into a condition expression:

1. Select the field you want to add a condition to.
2. Choose Tools | Tree to start the Tree Tool.
3. Choose Object | Branch. In the Object | Branch | Condition dialog box, choose Paste Function.

   An alphabetical list of function names appears.
4. Select the function name you want from the list.
5. Paste the name into the current expression by pressing *Enter* or by choosing OK.

   You can also double-click the function name you want to select and choose OK.

ObjectVision automatically inserts required left and right parentheses after the function name and places the pointer so that you can begin typing the arguments. You might want to check the Paste Arguments option to paste the function's required argument list. You can then replace the dummy arguments supplied by ObjectVision with the actual arguments. Any selected text is replaced with field or function names you paste next.

Defining expressions for special purposes

This section explains how you can define expressions to perform special functions such as:

- providing a default value for a field
- prompting a user for a field's value when other conclusions in the tree aren't applicable
- checking input values

Providing a default value

If a field will usually have a particular value, you can have ObjectVision calculate that value for the field as a convenience to the user. In the few cases when the field requires a different value, the user can override the calculated value. The Edit | Clear All and Form | Clear commands restore the default value for a calculated field. However, these commands also clear all values entered by a user.

You assign a default value for a field by defining a decision tree containing a conclusion node only. In the Conclusion dialog box, you enter the default value as a single constant or expression. ObjectVision displays that value in the field whenever the user opens your application.

For example, in the *Order* sample application, the decision tree for the Item field provides the default value Widget. The Item decision tree consists of a conclusion node with this value: *Widget*.

Users can enter a different value in the Item field if they want to unless you check No Override in the Field Protection dialog box.

Prompting for a calculated value

When you add a decision tree to a field, ObjectVision calculates the value of the field according to the tree logic. If none of the conclusions in the tree apply, ObjectVision is unable to calculate a value. However, a conclusion expression using *circular logic*—an expression that refers to its associated field—can prompt the user for a field's value when all other conclusions in the tree are inappropriate.

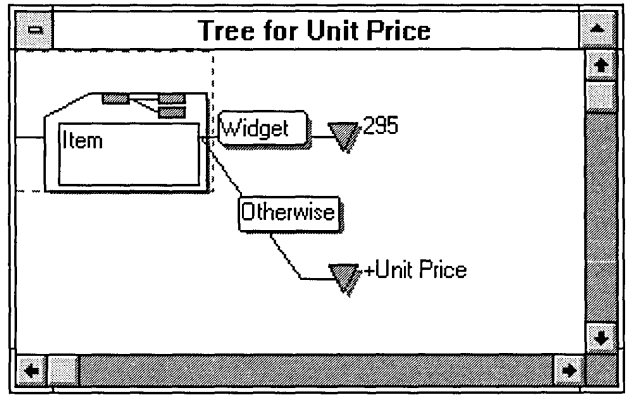
To provide a circular reference, you enter a conclusion with this value in the tree:

+Field Name

where *Field Name* is the name of the field for which you are defining the decision tree. Note that you must define the circular reference in the last conclusion of the tree.

For example, consider the Unit Price tree in the *Order* sample application, shown in Figure 7.6.

Figure 7.6
Decision tree for Unit Price



This tree tells ObjectVision to calculate the value 295 if the value of the Item field is the default, Widget. If the value of Item isn't Widget, the Unit Price tree prompts the user for the required Unit Price field's value.

Without circular logic, the Unit Price field would lack a decision tree, and the user would always have to enter the field's value—an inconvenience when the value is almost always the same.

Consider what would happen if the Unit Price decision tree didn't include the circular reference. As long as the Item field kept its default value, Widget, ObjectVision would be able to calculate a value for Unit Price. If the user overrode the Item value, however, ObjectVision would be unable to conclude a value for Unit Price, and the error result NA would appear in the field.

In this example, the circular logic is contained within one decision tree. You can also use circular logic in related decision trees. For example, suppose the tree for First includes a branch node Second, and the tree for Second includes a branch node First. Then, when a value is calculated for First, ObjectVision would try to calculate Second, prompting the user for First.

Checking input values

You can use the @functions @MESSAGE and @BLANK in decision trees to let users know when they have entered an incorrect value.

@MESSAGE displays error message strings you create and @BLANK removes the invalid value from the field. After @BLANK removes a value from a field, it is the next field in the browse sequence, so the user can enter a new value.

For example, you could add a decision tree to Approval Code in the *Credit* sample application to ensure that the user enters a three-digit value. Figure 7.7 shows how you might structure the tree.

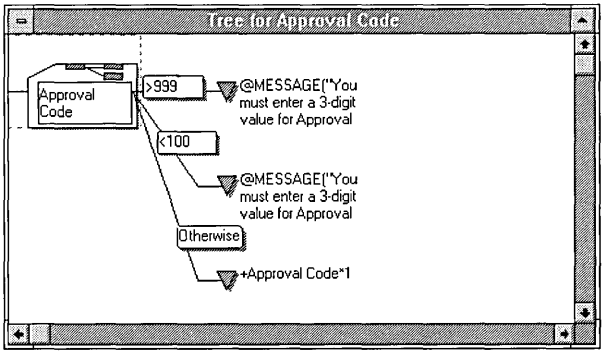


Figure 7.7
Decision tree for Approval Code

The conclusion nodes with @MESSAGE and @BLANK are shown truncated. The full expression is

```
@MESSAGE("You must enter a 3-digit value for Approval Code.")
@BLANK
```

C H A P T E R

8

@Function commands

This chapter has three main parts:

- "Using @functions" describes how to use @functions in decision tree expressions to calculate values. This section includes an explanation of how to enter different argument types (page 139), and a brief review of mathematical operators and their order of precedence (page 141).
- "Functions by type" lists @functions by category in Table 8.5, then briefly discusses each category. Most ObjectVision @functions are Quattro Pro-compatible, except for the link @functions and a few others unique to ObjectVision.
- "Function descriptions," beginning on page 147, is an alphabetical reference for each @function. Each entry includes format, arguments, a short description, and examples.

See Chapter 7 "Writing expressions," for more detailed information about entering and pasting @functions into decision trees.

Note that the examples given in this chapter assume your Windows Control Panel settings are the U.S. default settings.

Using @functions

You can use @functions in expressions within your decision trees to perform an operation on a field value or values. Depending on

where you put the function within an expression, the result of the function can be

- returned as the result of the expression
- combined with other data (using an operator)
- passed as an argument to another function

You can include functions as part of an expression using either the decision tree Condition dialog box or the decision tree Conclusion dialog box.

Function syntax

It's important to enter functions in the proper format, or *syntax*. All ObjectVision functions have the same basic syntax:

@FUNCTION(*Argument 1, Argument 2, ...*)

A function's syntax must meet the following rules:

- You must always include the leading @sign when typing a function name.
- You can type the function names in either uppercase or lowercase letters, or a combination of the two.
- You must enclose the required arguments following the function name in parentheses. You can omit parentheses following functions that need no arguments, such as @NOW, @NA, and @ERR; but ObjectVision accepts matching parentheses even when they aren't required.
- When there are multiple arguments, you must separate them with the separator character you specify in the Control Panel. In the U.S., the default list-separator character is the comma.
- If the function syntax specifies a certain order for arguments, you must enter them in that order.
- You cannot leave spaces between @ and the function name. You can leave spaces between arguments, parentheses, and the function name, but ObjectVision deletes them.
- You can "nest" functions inside other functions. For example, *@INT(@NOW)*.
- You can become more familiar with ObjectVision @functions and their required arguments by checking Paste Arguments in the Paste Function dialog box. This dialog box appears after you choose the Paste Function button in the Condition or Conclusion dialog box.

- You can also choose Help | @Functions in the Tree Tool to see information about the @functions and their arguments.

Function names

Each function has a unique name that helps identify the operation it performs. You can either type a function name into an expression or paste it in using the Condition | Paste Function or the Conclusion | Paste Function.

Pasting a function name avoids typographical errors and also provides an argument template for the function.

Arguments in functions

Argument is a generic term for the information required by the function. Most functions need at least one argument. The type of information required depends on the specific function. There are three general types of arguments:

- numeric values
- string values
- logical values

Each argument to a function can be a constant value, a field name, an expression, or another function.

When any of the function arguments are expressions or functions, ObjectVision first evaluates those arguments and passes the result of the evaluation to the outer function.

Enclose constants within quotation marks.

When you enter a constant string value as an argument, you must enclose it within double quotation marks. Constant string values include file names, and field names as arguments to link functions. Link names must also be enclosed within double-quotation marks.

When you enter a list of fields as an argument, you separate them with commas. It's necessary to enclose field names in single-quotation marks when the name contains a comma or an apostrophe. You must also enclose the entire comma-delimited list in double quotation marks.

However, when you use an ObjectVision field name as a reference to that field's value, you must omit the double quotation marks. For example, you can use the expression @LENGTH("*HMO*"), where *HMO* is a text constant, but you must use the expression

@LENGTH(*Medical Insurance Provider*) where *Medical Insurance Provider* is an ObjectVision field name.

When necessary, ObjectVision attempts to convert arguments to the type required by a function.

The following sections contain general information about each category of @functions. Refer to Table 8.5 for a list of all functions, and to the individual @function listings beginning on page 147 for details and examples.

Link functions

ObjectVision link functions connect to ASCII data files, Paradox, dBASE-compatible and Btrieve database files, and other Windows applications using the Dynamic Data Exchange (DDE) link mechanism. You must open a link to a data file before you can read from it, write to it, or change your location in it.

Each type of data file must be opened with the specific function for that type. You can close data files of all types by using the @CLOSE command. Most of the other link functions are used for moving to records, clearing records, or storing ObjectVision values in an external data file.

Mathematical functions

All mathematical functions perform calculations with numeric values as arguments and return a numeric value.

String functions

String functions manipulate character strings. When you calculate the position of a character in a string, use 0 for the first position, starting at the left.

Logical functions

Logical functions are used mostly in conditional statements, where the results are based on the validity of a logical expression such as *Quantity>0*. A conditional statement returns either the logical value *Yes* or the logical value *No*.

Date and time functions

Date and time functions calculate dates and times.

For calculation purposes, ObjectVision stores all dates as serial integers beginning with –36522 for January 1, 1800. The last valid serial integer is 73,050 for December 31, 2099. The way a date appears onscreen is a matter of formatting; you can choose a format for dates using Properties | Field Type in the Form Tool.

ObjectVision stores each expression of time as a decimal fraction where 0.000 represents 00:00:00, and 0.99999 represents 23:59:59. Again, you can format time expressions in your fields using Properties | Field Type in the Form Tool.

For example, @NOW returns the current date with the current time added as a fraction.

Miscellaneous functions

The miscellaneous functions perform error handling, value clearing, and value-type identification operations. Refer to the individual entries in the alphabetical lookup beginning on page 147.

Mathematical operators

The following two sections describe operator precedence and the use of operators.

Operator precedence

The result of an expression in an @function depends on the order in which ObjectVision performs the arithmetic operations. ObjectVision assigns each operator a precedence and performs the operations in order of precedence. For example, because multiplication has greater precedence than addition,

5 + 1 * 3 = 8, *not* 18

ObjectVision performs operations that have equal precedence from left to right.

Table 8.1 lists the operators allowed in ObjectVision expressions and the precedence assigned to each. The operator with the highest precedence number (that is, 7) is performed first.

Table 8.1
ObjectVision mathematical operators

Operator	Description	Precedence
&	String concatenation	1
= <>	Equal, not equal	2
< >	Less than, greater than	2
<=	Less than or equal	2
>=	Greater than or equal	2
- +	Subtraction, addition	3
* /	Multiplication, division	4
- +	Negative, positive	5
^	To the power of (exponential)	6
%	Percent	7

You can override the precedence of operators by including parentheses in your expression. Any operation enclosed in parentheses is given highest priority.

Operator use

You can freely intermix operators in mathematical, logical, and string @functions. Some operators act differently when used with strings rather than with numbers, as the following tables illustrate.

Arithmetic operators, such as +Order*1.3, calculate numeric values.

Table 8.2
Arithmetic operators

Operator	Use	Example
-	Negative or subtraction	@MOD(Unit,4)-@MOD(Unit2,4)
+	Positive or addition	@MOD(Unit,4)+Order
*	Multiplication	@ROUND(Price,0)*Unit
/	Division	@ABS(Carton/Unit)
^	Exponentiation	
%	Percent	

String operators act on strings of text. The & operator combines two strings into one. The relational operators (<, >, =, and so on) compare the alphabetical order of the characters that make up the strings. For this reason, +*"Bob"<"Carlos"* is *True* (that is, it has the value of Yes). Text operators do not distinguish between upper- and lowercase letters.

Table 8.3
String operators

Operator	Use	Example
&	Concatenation	@LENGTH(Address&"total")
<	Less than	@IF(State<3, "OK", " out of order")
>	Greater than	@IF(Insurance>(Salary*5), "out of order", "OK")
<=	Less than or equal to	@MID(MaritalStatus,1,1) <= "m"
>=	Greater than or equal to	+State >= "Florida"
< >	Not equal to	@LEFT(Residence,4) <> "New "
=	Equality	@IF(Marital Status = "single")

Logical operators act in true/false statements concerning values in fields; for example, +Quantity<10. ObjectVision checks the form to see if the statement is true (ObjectVision returns Yes for true and No for false).

Table 8.4
Logical operators

Operator	Use	Example
<	Less than	+500 < @MAX(RiskCategory, AgeGroup)
>	Greater than	+Credit > @SUM(Order1,Order2)
<=	Less than or equal to	@ABS(Age) <= 20
>=	Greater than or equal to	@ROUND(ShippingWeight,1) >= Minimum
< >	Not equal to	@SUM(Order1,Order2) <> 0
=	Equality	@IF(Carton=@MOD(Order1,ShipUnit), "verified", "try again")

Functions by type

Table 8.5
Functions listed by category

Function	Returns
Link functions	
@ASCIIOPEN(*LinkName, FileName, OVFields, Option*)	Opens the *LinkName* link between an ASCII file and the ObjectVision *OVFields*
@BOTTOM(*LinkName*)	Moves to the last record in the open database
@BTRVOPEN(*LinkName, Dictionary, TableName, DataFields, OVReadFields, OVWriteFields, IndexNum, Option*)	Opens the *LinkName* link the Btrieve table between *TableName DataFields* and the ObjectVision fields
@CLEAR(*LinkName*)	Clears the associated link fields in preparation for writing a new record
@CLOSE(*LinkName*)	Closes and disconnects the associated link fields and dissolves the link
@DBOPEN(*LinkName, FileName, DataFields, OVReadFields, OVWriteFields, IndexFile, Option*)	Opens a dBASE link between the dBASE data file's *DataFields* and ObjectVision's fields
@DDEOPEN(*LinkName, Application, Document, DataFields, OVFields*)	Opens a DDE link between *Application Document DataFields* and ObjectVision *OVFields*
@DELETE(*LinkName*)	Deletes current database record
@INSERT(*LinkName*)	Inserts the current Object-Vision values into the data file at the current record
@NEXT(*LinkName*)	Moves to the next data file record in the open link
@PREVIOUS(*LinkName*)	Moves to the previous record in the open database

Table 8.5: Functions listed by category (continued)

Function	Returns
Link functions (continued)	
@PXOPEN(*LinkName, FileName, DataFields, OVReadFields, OVWriteFields, SecIndexField, Option*)	Opens the *LinkName* link between the Paradox *Filename* *DataFields* and the ObjectVision fields
@STORE(*LinkName*)	Appends a new record or updates an existing record in the open database
@TOP(*LinkName*)	Moves to the first record in the open database or ASCII file
@UPDATE(*LinkName*)	Updates a record in the open database with the current ObjectVision values
Mathematical functions	
@ABS(*X*)	The absolute value of *X*
@INT(*X*)	The integer portion of *X*
@MAX(*List*)	The maximum value in *List*
@MIN(*List*)	The minimum value in *List*
@MOD(*X, Y*)	The remainder of *X*/*Y* (*Y*<>0)
@ROUND(*X, Num*)	*X* rounded to the number of digits specified with *Num* (up to 15)
@SUM(*List*)	The sum of values in *List*
String functions	
@CHAR(*Code*)	The ANSI character corresponding to code number *Code*
@CODE(*String*)	The ANSI code for the first character in *String*
@EXACT(*String1,String2*)	Yes if *String1* and *String2* are identical; otherwise, No
@FIND(*SubString, String, StartNumber*)	The character position of the first *SubString* found in *String*
@LEFT(*String, Num*)	The first *Num* characters in *String*
@LENGTH(*String*)	The number of characters in *String*
@LOWER(*String*)	*String* in lowercase letters
@MID(*String, StartNumber, Num*)	*Num* characters of *String*, beginning with the *StartNumber* character position
@REPEAT(*String, Num*)	*String*, repeated *Num* times

Table 8.5: Functions listed by category (continued)

Function	Returns
@REPLACE(*String, StartNum, Num, NewString*)	Removes *Num* characters from *String*, beginning with *StartNum*, then inserts *NewString* in its place
@RIGHT(*String, Num*)	The last *Num* characters in *String*
@UPPER(*String*)	*String* in uppercase characters
Logical functions	
@AND(*LogicalList*)	Yes if all arguments in *LogicalList* are true; otherwise, No
@IF(*Cond, TrueExpr, FalseExpr*)	*TrueExpr* if *Cond* is Yes, *FalseExpr* if *Cond* is No
@NOT(*Logical*)	Yes if *Logical* is false; otherwise, No
@OR(*LogicalList*)	Yes if any argument in *LogicalList* is true; otherwise, No
Date and time functions	
@DATE(*Yr, Mo, Day*)	A date serial number
@DATEVALUE(*DateString*)	A date serial number
@DAY(*DateTimeNumber*)	The day of the month (1-31)
@HOUR(*DateTimeNumber*)	The hour of the day (0-23)
@MINUTE(*DateTimeNumber*)	The minute of the hour (0-59)
@MONTH(*DateTimeNumber*)	A month number (1-12)
@NOW	The current date/time serial number
@SECOND(*DateTimeNumber*)	A second number (0-59)
@TIME(*Hr, Min, Sec*)	A time serial number
@TIMEVALUE(*TimeString*)	A time serial number
@WEEKDAY(*DateTimeNumber*)	The day of the week (1-7)
@YEAR(*DateTimeNumber*)	A year number (1800-2099)
Miscellaneous functions	
@BLANK	Assigns a blank value to a field
@ERR	The value ERR (error)
@MESSAGE(*String*)	Displays *String* in a window with an OK button
@NA	The value NA (Not Available)
@TYPE(*Value*)	A number indicating the data type of *Value* (1,2,4, or 16)

Function descriptions

This section lists ObjectVision @functions in alphabetical order. It shows format, argument requirements, and examples for each.

To see brief descriptions of each @function listed by type of function, refer to Table 8.5, beginning on page 144.

ABS

Format @ABS(*X*)

X = a numeric value

Use

@ABS returns the absolute (positive) value of *X*.

Examples

@ABS(Actual Weight – Ideal Weight) = 7, where Actual Weight – Ideal Weight is –7 or 7

@ABS(-100) = 100
@ABS(100) = 100
@ABS(0) = 0

AND

Format @AND(*LogicalList*)

LogicalList = one or more logical values or expressions that evaluate to a logical value, up to a maximum of 14

Use

Returns the logical value Yes if all of the arguments evaluate as true; otherwise, it returns the logical value No.

Examples

@AND(Qualified, Citizen) = Yes, where the values of both the Qualified and Citizen fields are Yes; otherwise, it returns No.

ASCIIOPEN

Format @ASCIIOPEN(*LinkName, FileName, OVFields, Option*)

LinkName = the name you specify for the link and use to reference the link in all subsequent functions.

FileName = the name of an ASCII file in the current directory. When the ASCII file is not in the current directory, you can use its complete pathname. Data values within the file must be delimited by commas. Values that contain commas or double quotes must be surrounded by double quotes. Other values can optionally be surrounded by double quotes.

If there are fewer values on the line than there are linked ObjectVision fields, null values are assumed for any remaining ObjectVision fields linked to the file. If there are too many values on the line, the extra values at the end are skipped.

OVFields = a comma-delimited list of ObjectVision field names.

Option = READ, WRITE, or APPEND, depending on the intended use of the file.

Use

- **READ:** The file is opened for reading, and the first line of values is imported into the ObjectVision fields listed in *OVFields*.
- **WRITE:** The file is opened for writing at the beginning of the file. Any data already in the file is lost. If the file does not exist, it is created.
- **APPEND:** The file is opened for writing at the end of the file. If the file does not exist, it is created.

@ASCIIOPEN opens the ASCII file *FileName* and links it with the ObjectVision fields in *OVFields*.

@ASCIIOPEN returns Yes if successful; otherwise, No.

Examples

@ASCIIOPEN("CURREAD","CURSTOCK.ASC ","High,Low,Buy,Sell", "READ") This function opens the CURREAD link to

the ASCII file CURSTOCK.ASC and reads in values for the ObjectVision fields High, Low, Buy, and Sell.

BLANK

Format @BLANK

Use

@BLANK assigns a blank (null) value to a field, letting you create a method to repeatedly prompt the user after entry of an invalid value.

You can use this function in combination with the @MESSAGE function, which displays any string message given as an argument.

Examples

@MESSAGE(Quantity &" is an invalid value for Quantity. Enter a positive value")
@BLANK

This example displays a message before removing an invalid value entered in the Quantity field.

BOTTOM

Format @BOTTOM(*LinkName*)

LinkName = a database file link name

Use

@BOTTOM repositions the *LinkName* database link to the last record in the Paradox, dBASE, or Btrieve database file based on the link indexing order. *LinkName* must be the name of a database link previously opened with the Tools | Link command, or the @PXOPEN, @DBOPEN, or @BTRVOPEN function.

@BOTTOM uses the database order specified by the link. Returns Yes if successful; otherwise, No.

Examples

@BOTTOM("ORDER"), as used in the *Status* sample application to reposition the write link to the last record in the ORDER database.

BTRVOPEN

Format @BTRVOPEN(*LinkName, Dictionary, TableName, DataFields, OVReadFields, OVWriteFields, IndexNum, Option*)

LinkName = the name you specify for the link and use to reference the link in all subsequent functions.

Dictionary = the path name of the XQL files (FIELD.DDF, FILE.DDF, and INDEX.DDF) that contain the field definitions for *TableName*.

TableName = the name of a valid Btrieve table name contained in Dictionary. When the file is not in the current directory, you can use its complete pathname.

DataFields = contains the names of the database fields, delimited by commas, to be linked to your ObjectVision fields.

OVReadFields = contains the names of your ObjectVision fields, delimited by commas, to hold imported values from *FileName DataFields*. There must be a one-to-one correspondence between the database file and your ObjectVision fields.

OVWriteFields = contains the names of your ObjectVision fields, delimited by commas, to write their values to *FileName DataFields*. There must be a one-to-one correspondence between the database file and your ObjectVision fields.

IndexNum = specifies which predefined indexing identifier to use (an existing ID from 0 through 23) as defined in the table.

Option = specifies how closely you want to match the index value.

Use

- **INEXACT:** If *Option* is INEXACT, the record closest to the index value is located.

- **"" (BLANK):** If *Option* is blank (""), only records identical to the index value are located.

@BTRVOPEN opens the Btrieve file identified by the XQL table *TableName* and links it with the ObjectVision fields. @BTRVOPEN returns Yes if successful; otherwise, No.

Btrieve links require a dictionary that defines field names, locations, sizes, and types for fields in the Btrieve file. The dictionary also defines indexing schemes for accessing records based on the values of certain index fields.

ObjectVision uses the same dictionary structure Novell XQL uses. If you are linking to a pre-existing Btrieve file, use XQL or other Novell products to create a dictionary if one does not exist.

Examples

@BTRVOPEN("Patients", "C:\Btrieve", "Patients", "ID, First Name", "ID, First", "ID, First", "0","")

CHAR

Format @CHAR(*Code*)

Code = a numeric value from 1 to 255, inclusive

Use

@CHAR returns the onscreen character corresponding to the given code. Refer to Appendix C, "The ANSI character set," for a complete list of characters and their corresponding codes.

Examples

@CHAR(33) = !
@CHAR(34) = "
@CHAR(35) = #
@CHAR(36) = $
@CHAR(75) = K

CLEAR

Format @CLEAR(*LinkName*)

LinkName = a Paradox, dBASE, or Btrieve database file link name or an ASCII file name

Use

@CLEAR clears the ASCII, Paradox, dBASE, or Btrieve link fields associated with the *LinkName* link in preparation for writing a new record. Because the link position is undefined after @CLEAR is carried out, an error message appears if either @PREVIOUS or @NEXT is executed afterward.

LinkName must be the name of a link previously opened with Tools | Links or @ASCIIOPEN, @PXOPEN, @DBOPEN, @BTRVOPEN.

You might place a button field in your applications' forms to make it easy for the user to clear the form of linked values.

@CLEAR returns Yes is successful; otherwise, No.

Examples

@CLEAR("ORDER"), as used in the *Credit* sample application. This function clears the fields in the ORDER.DB database through the open database link ORDER.

CLOSE

Format @CLOSE(*LinkName*)

LinkName = an ASCII, Paradox, dBASE, Btrieve, or DDE data file link name

Use

@CLOSE disconnects the linked fields, clears the ObjectVision values, and dissolves the *LinkName* link.

LinkName must be the name of a link previously opened with the Tools | Links command or the @ASCIIOPEN, @PXOPEN, @DBOPEN, @BTRVOPEN, or @DDEOPEN functions. Any fields containing values unrelated to links retain their current values.

@CLOSE returns Yes if successful; otherwise, No.

Examples

@CLOSE("CURSTOCK.ASC"), closes the ASCII link CURSTOCK.ASC and unlinks it from the associated ObjectVision fields.

@CLOSE("ORDER"), as used in the *Credit* sample application, unlinks the ORDER.DB database from the associated ObjectVision fields using the database link ORDER.

CODE

Format @CODE(*String*)

String = a string value

Use

@CODE returns the ANSI code of the first character in *String*. This is the opposite of the @CHAR function, which returns the character corresponding to the given code. If *String* is the empty string, ERR is returned.

Examples

@CODE("!") = 33
@CODE("#") = 35
@CODE("$") = 36
@CODE("?") = 63
@CODE(Hello) = 72 (When Hello is a field name with a value = Handicapped)
@CODE("Sam") = 83 (code for *S*)
@CODE(Marital Status) = 87, (*W*, when the field value = Widowed)

DATE

Format @DATE(*Yr,Mo,Day*)

Yr = a numeric value from 1800 to 2099 (or from 0 to 199 for Quattro Pro compatability)
Mo = a numeric value from 1 to 12
Day = a numeric value from 1 to 31

Use

@DATE returns the serial number of the date specified with year, month, and day arguments. This serial number can range from –36522 to 73050, and represents dates from January 1, 1800 to December 31, 2099. Serial number 0 represents December 30, 1899.

The fractional portion of a date serial number is used by the time functions. The *Mo* and *Day* values can be outside of the above range to represent displacements from the current month or day.

Examples

@DATE(1987,1,1) = 31778
@DATE(1987,9,13) = 32033
@DATE(1900,1,1) = 2
@DATE(1991,6,19 – Lead Time) = 33394, (representing June 5, 1991 where the value of the Lead Time field is 14).

DATEVALUE

Format @DATEVALUE(*DateString*)

DateString = a string value in any valid date format, enclosed by quotes

Use

@DATEVALUE returns a serial date value that corresponds to the value in *DateString*. This serial number can range from –36522 to 73050, and represents the number of days from December 30, 1899 up to the date referenced in the expression. The earliest date available is January 1, 1800.

If the value in *DateString* is in an incorrect format or represents an impossible date, an ERR value is returned.

The same date formats users can enter in a field can be used to enter the *DateString* argument. The / (slash) is interchangable with the – (hyphen) when you enter the *DateString*. Trailing or leading spaces are ignored. There are five valid formats for *DateString*, assuming U.S. defaults have been set in the Windows Control Panel:

- DD-MMM-YY ("04-Jul-87")
- DD-MMM ("04-Jul") (assumes the current year)
- MMM-YY ("Jul-87") (assumes the first of the month)
- MM/DD/YY ("11/9/95") (the short date format in the Windows Control Panel)
- MMMM D, YYYY ("January 1, 1992") (the long date format in the Windows Control Panel)

Examples

@DATEVALUE("07/04/87") = 31962

@DATEVALUE("JUL-86") = 31594
@DATEVALUE("July 1, 1986") = 31594
@DATEVALUE("07/18/87") = 31976
@DATEVALUE("04-may-87") = 31901
@DATEVALUE(07/04/87) = ERR (absence of quotes makes ObjectVision divide the numbers)
@DATEVALUE("May-1987") = 31898

DAY

Format @DAY(*DateTimeNumber*)

DateTimeNumber = a number in the range –36522 (January 1, 1800) to 73050 (December 31, 2099)

Use

Converts the date/time serial number you supply as *DateTimeNumber* into the number associated with that day (1-31). Decimal portions of *DateTimeNumber* represent the time from 12:00 a.m. to 11:59:59 p.m. and are computed as a fraction of a 24-hour day.

Examples

@DAY(31779) = 2 (1/2/87)
@DAY(32134) = 23 (12/23/87)
@DAY(@DATE(1987,9,10)) = 10
@DAY(73051) = ERR because the number you entered was larger than 73050
@DAY(Date) = 1, when the Date field value is the date/time serial number for 1/Feb/1990

DBOPEN

Format @DBOPEN(*LinkName, FileName, DataFields, OVReadFields, OVWriteFields, IndexFile, Option*)

LinkName	=	name you specify for the dBASE-compatible database link.
FileName	=	name of the dBASE-compatible database file (.DBF).
DataFields	=	names of the database fields, delimited by commas, to be linked to your ObjectVision fields.

OVReadFields	=	names of your ObjectVision fields, delimited by commas, to hold imported values from *FileName DataFields*. There must be a one-to-one correspondence between the database and your ObjectVision fields.
OVWriteFields	=	names of your ObjectVision fields, delimited by commas, to write their values to *FileName DataFields*. There must be a one-to-one correspondence between the database and your ObjectVision fields.
IndexFile	=	(optional) name of a dBASE-compatible index file (.NDX). The .NDX file must be in the same directory as the .DBF database file unless you specify the complete pathname. ObjectVision uses the index file, if present, to order and locate the database records.
Option	=	(optional) INEXACT indicates that you want to locate the record which has the same beginning characters as the index value. If blank (""), the index value must match exactly when locating records.

IndexFile and *Option* are both optional arguments, but their relative positions *must* be maintained. Use double quotation marks to indicate blank values for these arguments.

Use

@DBOPEN opens a named, bi-directional database link between *DataFile* (a dBASE file) and your ObjectVision fields.

@DBOPEN returns Yes if successful; otherwise, No.

Examples

```
@DBOPEN("DBORDER","DBORDER.DBF", "Term,Auth","Credit
        Terms,Credit Authorization","Credit Terms,Credit
        Authorization","DBORDER.NDX","")
```

This example opens a write link, named DBORDER, to the DBORDER.DBF database. The database fields are Term and Auth, and the ObjectVision fields are Credit Terms and Credit Authorization.

DDEOPEN

Format @DDEOPEN(*LinkName, Application, Document, DataFields, OVFields*)

LinkName = name you specify for the DDE link.

Application = name of a Windows application.

Document = name of a document file to be referenced in the Windows *Application*. If no Windows application contains this document, ObjectVision will attempt to start the Windows application with the *Document* name.

DataFields = names of the data fields in the Windows *Document*, delimited by commas. There must be a one-to-one correspondence between the DDE *DataFields* and your ObjectVision *OVFields*.

OVFields = names of the ObjectVision fields, delimited by commas, that you want to link to the Windows *DataFields*. There must be a one-to-one correspondence between the DDE *DataFields* and the ObjectVision fields.

Use

@DDEOPEN opens a named, directional DDE link between a Windows *Application* and your ObjectVision fields.

@DDEOPEN returns Yes if successful; otherwise, No.

ObjectVision automatically displays new values for the linked fields whenever data values change in the linked Windows document.

Examples

```
@DDEOPEN("INTLTAX","Excel", "EXCISE.XLS","Country,Curtax",
        "Country of Origin,Current Tax")
```

This example opens a DDE link named INTLTAX to the Windows application Excel and the document file EXCISE.XLS. The Excel data fields are Country and Curtax, and the ObjectVision fields are Country of Origin and Current Tax.

DELETE

Format @DELETE(*LinkName*)

LinkName = an open link to a database file

Use

@DELETE deletes the record at the current location in the database file using the *LinkName* link. ObjectVision is unable to display a record which has been deleted (or marked for deletion, in the case of dBASE databases).

LinkName must be the name of a database link previously opened with the Tools | Link command, or the @PXOPEN, @DBOPEN, or @BTRVOPEN function.

@DELETE returns Yes if successful; otherwise, No.

Examples

@DELETE("ORDER"), as used in the *Credit* sample application. This function marks for deletion the current record in the ORDER.DB database using the ORDER link.

ERR

Format @ERR

@ERR returns the error value ERR in the current field and in any other fields that reference the current field, either directly or indirectly.

The ERR value resulting from this function is used most often with the @IF function, to bring attention to error conditions.

Examples

@ERR = ERR
@IF(Field1>Field2,0,@ERR) = 0 (if Field1>Field2) or ERR (if Field1<=Field2)

EXACT

Format @EXACT(*String1*, *String2*)

String1 = a valid string value
String2 = a valid string value

Use

@EXACT compares the values of *String1* and *String2*. If the values are *exactly* identical, including capitalization and diacritical marks (such as ~), it returns Yes. If there are any differences, it returns No.

To compare strings or cell contents without regard to capitalization or diacritical marks, use the = operator. For example, *+Field1=Field2* returns *Yes* if the contents of the fields are the same but are capitalized differently.

Examples

@EXACT("client","Client") = No
@EXACT("client","client") = Yes
@EXACT(29,"29") = Yes
@EXACT(Field1,"yes") = Yes (if Field1 contains the string value *yes*)
@EXACT(client,client) = Yes

FIND

Format @FIND(*Substring*, *String*, *StartNumber*)

SubString = a valid string value, representing the value to search for

String = a valid string value, representing the value to search through

StartNumber = a numeric value ≥ 0, representing the character position at which to begin searching

Use

@FIND searches through the given *String* for the value given as *Substring*. If it finds *Substring*, it returns the character position at which the first occurrence was found. *StartNumber* begins the search at that number of characters into the string. 0 = the first character in the string, 1 = the second, and so on.

@FIND is case-sensitive and is also sensitive to diacritical marks used in non-English languages. You can overcome the case-sensitivity of this function by using @UPPER to force one or more of the strings into all caps; for example,

@FIND(@UPPER(FieldA),@UPPER(FieldB),0)

forces both the substring in FieldA and the string in FieldB to uppercase, then searches for the substring.

@FIND is most often used in conjunction with two other string functions: @REPLACE (to perform search-and-replace operations on strings) and @MID (to access substrings).

If @FIND fails to find any occurrences of *Substring*, or if the *StartNumber* given is less than 0 or greater than the length of *String* the result is ERR.

Examples

@FIND("i","find",0) = 1
@FIND("nd","find",2) = 2
@FIND("F","find",0) = ERR
@FIND("f","find",3) = ERR
@FIND("d","find",4) = ERR
@FIND("hi",FieldM,0) = 1 (if the value of FieldM is *ship*)
@FIND("e",Rating,5) = 3 (if the value of Rating is *Excellent*)

HOUR

Format @HOUR(*DateTimeNumber*)

DateTimeNumber = a numeric value from –35522 to 73050.99999, representing a date/time serial number

Use

@HOUR returns the hour portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/time serial number. Because only the decimal portion of a serial number evaluates to a time, the integer portion of the number is disregarded. The result is a number from 0 (12:00 a.m.) to 23 (11:00 p.m.).

To return standard hours (1–12) instead of military hours (1–24), use the @MOD function (page 167) with a parameter of 12.

(See also @TIME on page 177 and @TIMEVALUE on page 177.)

Examples

@HOUR(.25) = 6
@HOUR(.5) = 12
@HOUR(.75) = 18
@HOUR(@TIMEVALUE("10:08am")) = 10
@MOD(@HOUR(@TIMEVALUE("9:31:52 PM")),12) = 9
@HOUR(Time of Occurrence) = 11, when the Time of Occurrence field value is 11:30 a.m.

IF

Format @IF(*Cond, TrueExpr, FalseExpr*)

Cond = a logical expression representing the condition to be tested

TrueExpr = a value representing the value to return if *Cond* is true

FalseExpr = a value representing the value to return if *Cond* if false

Use

@IF evaluates the logical condition given as *Cond.* If the condition is found to be true, @IF returns the value given as *TrueExp.* If the condition is false, @IF returns the value given as *FalseExpr.*

The expression entered as *Cond* can be any logical expression that can be evaluated as true or false; for example, *Order<0* or *Quantity*Price=53.*

You can use compound conditions by connecting expressions with @AND or @OR. If you use @AND, all conditions given must be met for the compound condition to evaluate to true. If you use @OR, the expression is true if any of the conditions is met. For example, @AND(*Quantity<50, Quantity>5*) means that the value in Quantity must be between 5 and 50 to evaluate true.

You can also use the @NOT operator to negate a condition. For example, @NOT(*Age>65*) evaluates true if the value of field Age is *not* greater than 65.

The expressions given as *TrueExpr* and *FalseExpr* can be any valid expression. If either *Cond, TrueExpr,* or *FalseExpr* is an error value (NA or ERR), @IF returns that error value.

@IF functions can be nested, or used within one another. In other words, *TrueExpr* can contain yet another test to further validate *Cond*. There's no limit on the number of @IF expressions that you can nest, as long as the entire expression doesn't exceed 4096 characters.

Examples

@IF(8=7,4,5) = 5
@IF(Order<100,"Yes","No") = Yes if Order < 100; otherwise, No
@IF(Amount Requested>Amount Allowed, "Ms. Andrews ","") = Ms. Andrews if the value of the Amount Requested field is greater than value of the Amount Allowed field; otherwise, the result is "" (the empty string).

INSERT

Format @INSERT(*LinkName*)

LinkName = an open link name

Use

@INSERT writes associated ObjectVision field values to a new record for an ASCII, Paradox, dBASE, or Btrieve link.

FileName must be the name of a file previously opened with Tools | Links, or the @ASCIIOPEN, @PXOPEN, @DBOPEN, or @BTRVOPEN function. ObjectVision fields linked to the file maintain their current values.

All values written to an ASCII file are double-quoted and comma delimited. Each double quote in a value will be represented with two double quotes. An end-of-line character is written after the last value when inserting records using ASCII links.

The insert fails if any database constraints are violated by the insertion. For example, if the record violates a key within a Paradox table a message appears stating that the insert will fail.

@INSERT returns Yes if successful; otherwise, No.

Examples

@INSERT("BUYRECS") writes the associated ObjectVision field values into the BUYRECS link.

INT

Format @INT(*X*)

X = a numeric value

Use

@INT drops any fractional portion of *X*, returning only the integer value. See @ROUND (page 174) for a function that rounds *X* to the nearest integer.

Examples

@INT(499.99) = 499
@INT(0.1245) = 0
@INT(Lowest Score) = 6, if the Lowest Score field value is 6.9
@INT(Credit Rating) = –76, if the Credit Rating field value is –76.9

LEFT

Format @LEFT(*String, Num*)

String = a string value
Num = a numeric value ≥ 0

Use

@LEFT returns the leftmost *Num* characters of *String*. It lets you extract a specified number of characters from the left side of a string.

If *Num* is longer than the length of *String*, all of *String* is returned.

Examples

@LEFT("Jennifer",5) = Jenni
@LEFT("Jennifer",15) = Jennifer
@LEFT("155",1) = 1
@LEFT(" Jennifer",6) = J (including five leading spaces)
@LEFT(Job Grade,5) = Super, where the Job Grade field value is Supervisor

LENGTH

Format @LENGTH(*String*)

String = a string value

Use

@LENGTH returns the number of characters in *String*, including spaces. You can combine strings with the string concatenation operator, an ampersand (&). When *String* is a text string, it must be enclosed by double quotes.

Examples

@LENGTH("Hello, world.") = 13
@LENGTH(" Jennifer") = 9 (including preceding space)
@LENGTH("Greetings "&"earthling") = 19 (including space after Greetings)
@LENGTH(FieldA&FieldB) = total number of characters in FieldA and FieldB
@LENGTH(Medical Insurance Provider) = 3, when the Medical Insurance Provider field value is HMO

LOWER

Format @LOWER(*String*)

String = a string value

Use

@LOWER returns *String* in lowercase characters. Numbers and symbols within a string are unaffected.

Examples

@LOWER("UPPER") = upper
@LOWER("Hello, world.") = hello, world.
@LOWER("145 Bancroft Lane") = 145 bancroft lane
@LOWER(4839) = 4839
@LOWER(@LEFT("Johnson",1)) = j
@LOWER(First Name) = joanne, when the First Name field value is JoAnne

MAX

Format @MAX(*List*)

List = two or more numeric values up to a maximum of 14

Use

@MAX returns the largest numeric or date value in *List*. Text values are converted to numbers, if possible. If ObjectVision is unable to convert any of the arguments to a number, or if any of the fields referenced contain ERR, the resulting value is ERR. Logical values are converted to 1 (Yes) or 0 (No).

Examples

@MAX(Risk Category, Age Group) = 8, where the Risk Category field value is 8 and the Age Group field value is 5.

MESSAGE

Format @MESSAGE(*String*)

String = a string value

Use

@MESSAGE displays *String* in a message box with an OK button.

You typically use this function to inform the user that the value just entered is invalid, such as an out-of-range value. @MESSAGE is often combined with the @BLANK function, which assigns a blank value to the current field.

Examples

@MESSAGE("You must enter a value with at least 5 digits ")
@BLANK

When the user enters an out-of-range value in the Zip Code field, the String appears and the field value is cleared to let the user enter a correct value. In the decision tree for Zip Code, you can test for the condition <5, and put this example in the conclusion node.

MID

Format @MID(*String,StartNumber,Num*)

String = a string value
StartNumber = a numeric value ≥ 0
Num = a numeric value ≥ 0

Use

@MID extracts the first *Num* characters of *String* starting at character number *StartNumber*. It is similar to the @LEFT function, which extracts *Num* characters of *String* beginning with the first character. The difference is that you can specify the character number to start with. Note that the first character of a string is character 0.

If *StartNumber* is greater than the length of *String*, or if *Num* is 0, the result is "", or an empty string.

Examples

@MID("Abraham Lincoln",8,7) = Lincoln
@MID("George Washington",7,4) = Wash
@MID("Theodore Roosevelt",19,5) = ""
@MID(ZIP Code,0,3) = 787, when the ZIP Code field value is 78751
@MID(President,@FIND("Roosevelt",President,0),
@LENGTH("Roosevelt")) = Roosevelt (if the President field value is Franklin Roosevelt)

MIN

Format @MIN(*List*)

List = two or more numeric values up to a maximum of 14

Use

@MIN returns the smallest numeric value in *List*. Text values are converted to numbers, if possible, and logical values are converted to 1 (Yes) or 0 (No).

@MIN returns ERR if ObjectVision is unable to convert any argument to a number. If a field has a blank value (""), it is assumed to be 0.

Examples

@MIN(Risk Category,Age Group) = 5, when the Risk Category field value is 8 and the Age Group field value is 5

MINUTE

Format @MINUTE(*DateTimeNumber*)

DateTimeNumber = a numeric value from –36522 to 73050, representing a date/time serial number

Use

@MINUTE returns the minute portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/time serial number. If *DateTimeNumber* is given as text, it is converted to a serial number. Because only the decimal portion of a serial number evaluates to a time, the integer portion of the number is disregarded. The result is a number from 0 to 59.

To extract the minute portion of a string that is in time format (instead of serial format), use @TIMEVALUE within the @MINUTE function to translate the time into a serial number. You can also use @TIME to enter a time value instead of a serial number.

Examples

@MINUTE(Time of Occurrence) = 30, when the Time of Occurrence field value is 11:30 a.m.

MOD

Format @MOD(*X*,*Y*)

X = a numeric value
Y = a numeric value ≠ 0

Use

@MOD divides the *X* value by *Y* and returns the modulus, or remainder value. The result has the same sign as *Y*. Because you cannot divide a number by zero, ERR results if the value of *Y* is zero.

Examples

@MOD(3,1) = 0 (3 divided by 1 leaves no remainder)
@MOD(5,2) = 1 (5 divided by 2 leaves a remainder of 1)
@MOD(3,1.1) = 0.8
@MOD(4,0) = ERR

@MOD(Field Kits,Field Kit Orders) = 50, when the Field Kits field value is 500 and the Field Kit Orders field value is 450

MONTH

Format @MONTH(*DateTimeNumber*)

DateTimeNumber = a numeric value from –36522 to 73050, representing a date serial number

Use

@MONTH returns the month portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/time serial number. Only the integer portion is used. The result is an integer value from 1 (January) to 12 (December).

To extract the month portion of a string that is in date format (instead of serial format), use @DATEVALUE within the @MONTH function to translate the date into a serial number (see page 154). You can also use @DATE to enter a date value instead of a serial number (see page 153).

Examples

@MONTH(69858) = 4
@MONTH(58494) = 2
@MONTH(.37373) = 12
@MONTH(@DATEVALUE("3/5/88")) = 3
@MONTH(@DATE(1988,3,5)) = 3
@MOD(@MONTH(@DATEVALUE("3/5/88")),12) = 3
@MONTH(Date) = 2, when the Date field value is 1/Feb/1991

NA

Format @NA

@NA returns the special value NA (Not Available). Expressions that depend on a value entered as @NA return the value NA, unless there is an error, in which case they return ERR.

@NA is used to indicate values not included as a possible conclusion in a decision tree. @NA ensures that inaccurate data is not displayed when ObjectVision evaluates an unexpected value.

Examples

@NA = NA
@IF(Orders<0,@NA,Orders) = NA, when the Orders field is less than 0; otherwise, the value of Orders

NEXT

Format @NEXT(*LinkName*)

LinkName = an open ASCII, Paradox, dBASE, or Btrieve link name

Use

@NEXT reads the next record of a database link or the next line of an ASCII link.

@NEXT returns Yes if successful; otherwise, No.

Example

@NEXT("ORDER") = Yes, when the open database link ORDER locates the next record in the ORDER.DB file
@NEXT("ORDER") = No, when the database link ORDER is disconnected

NOT

Format @NOT(*Logical*)

Logical = a value of Yes or No

Use

@NOT inverts the value of *Logical* and returns Yes if *Logical* is No; otherwise, No if *Logical* is Yes. *Logical* could be a much more complex logical expression than this example.

Example

@NOT(Smoker) = Yes, when the Smoker field value is No

NOW

Format @NOW

Use

@NOW returns the serial number of the current date and time in the user's computer clock. The value generated by @NOW is updated to the current date and time each time an expression recalculates a field value.

This serial number can range from –36522 to 73050, and represents the number of days from December 30, 1899 up to the current date. The earliest date available is January 1, 1800 and the latest date available is December 31, 2099.

The integer part of a date/time serial number evaluates to the date; the decimal portion evaluates to the time. To extract just the date portion, use @INT(@NOW). To extract just the time portion, use @MOD(@NOW,1).

Examples

@NOW = 31905.572338 (5/8/87, 1:45 PM)
@INT(@NOW) = 31905 (5/8/87)
@MOD(@NOW,1) = 0.572338 (1:45 PM)
@INT(@MOD(@NOW,7)) = 6 (the number of the day of the week)

OR

Format @OR(*LogicalList*)

LogicalList = one or more logical values or expressions that evaluate to a logical value, a value of Yes or No, up to a maximum of 14.

Use

@OR returns Yes if any argument in *LogicalList* is Yes; otherwise, No.

Examples

@OR(Self-Employed,Retired) = Yes, when either the Self-Employed field value or the Retired field value is No

@OR(Self-Employed,Retired) = No, when both the Self-Employed field value and the Retired field value is Yes

PREVIOUS

Format @PREVIOUS(*LinkName*)

LinkName = the name of the open Paradox, dBASE, or Btrieve database link

Use

@PREVIOUS reads the record just before the current database record as sorted by link index order.

@PREVIOUS returns Yes if successful; otherwise, No.

Example

@PREVIOUS("ORDER") = Yes, when the open database link INVEN locates the next record in the INVENTOR.DB file
@PREVIOUS("ORDER") = No, when the database link INVEN is disconnected

PXOPEN

Format @PXOPEN(*LinkName, TableName, DataFields, OVReadFields, OVWriteFields, SecIndexField, Option*)

LinkName = name you specify for the Paradox database file link.

TableName = name of the Paradox database table (.DB).

DataFields = names of the database fields, delimited by commas, to be linked to your ObjectVision fields.

OVReadFields = names of your ObjectVision fields, delimited by commas, to hold imported values from *TableName DataFields*. There must be a one-to-one correspondence between the database and your ObjectVision fields.

OVWriteFields = names of your ObjectVision fields, delimited by commas, to write their values to *TableName DataFields*. There must be a one-to-one correspondence between the database and your ObjectVision fields.

SecIndexField = (optional) name of the *TableName* field to be used as the index for database access. This

field, if specified, is used instead of the database's primary index to order and locate records. If blank (""), the primary index is used.

Option = (optional) INEXACT indicates that you want to locate the record which is closest to the index value. If blank (""), the index value must match exactly when locating records.

SecIndexField and *Option* are both optional arguments, but their relative positions must be maintained. Use double quotation marks to indicate blank values for these arguments.

Use

@PXOPEN opens a named, bi-directional database link between *DataFile* and your ObjectVision fields.

@PXOPEN returns Yes if successful; otherwise, No.

Examples

```
@PXOPEN("INVEN","INVENTOR", "Item,Quantity",
        "Warehouse,City", "Warehouse,City","", "")
```

This function opens the INVEN write link to the Paradox INVENTOR.DB data file. The database fields are Item and Quantity, and the ObjectVision fields are Warehouse and City.

REPEAT

Format @REPEAT(*String,Num*)

String = a string value
Num = a numeric value ≥ 0

Use

@REPEAT returns *Num* copies of *String* as one continuous string. You can specify exactly how many times you want the string to be repeated. The @REPEAT function displays a fixed number of copies of *String*.

If *Num* equals 0, @REPEAT returns "". @REPEAT returns ERR (error) if *Num* is less than 0.

When you specify a text string with @REPEAT, it must be enclosed by double quotes.

Examples

@REPEAT("-",20) = --------------------
@REPEAT("good day!",3) = good day!good day!good day!
@REPEAT(City,5) = the City field value repeated 5 times
@REPEAT("-",@LENGTH(Jane Hopper)) = ----------

REPLACE

Format @REPLACE(*String, StartNum, Num, NewString*)

String	=	a string value, representing the text to operate on
StartNum	=	a numeric value ≥ 0, representing the character position to begin with
Num	=	a numeric value ≥ 0, representing the number of characters to delete
NewString	=	a string value, representing the characters to insert at position *Num*

Use

@REPLACE lets you replace characters in a text string with a new string. It searches through the given *String* until it reaches the given character position *StartNum*. Then it removes *Num* number of characters from the string, replacing them with *NewString*.

Note that for @REPLACE, as for other string functions, the first character of *String* is counted as character 0.

Both *String* and *NewString* can be either references to field values or text strings. If text strings, they must be enclosed by double quotes.

To replace one string with another, specify 0 as *StartNum*. For *Num*, enter a number equal to or greater than the number of characters in *String*.

To insert one string into another string, specify 0 as *Num*.

To add one string to the end of another, specify as *StartNum* a number equal to the number of characters in *String*.

To delete part or all of a string, specify "" as *NewString*.

Examples

@REPLACE("McDonald Corp.",2,6,"Douglas") = McDouglas Corp.

@REPLACE("Leslie J. Cooper",7,3,"") = Leslie Cooper
@REPLACE("Sales Salaries",6,0, "Reps' ") = Sales Reps' Salaries
(There must be a space between Reps and the final quotation mark)
@REPLACE("355 Howard",9,0," St.") = 355 Howard St. (There must be a space between " and *St.*)
@REPLACE(Distribution Area,30,100," USA only") = Limited USA only, when the Distribution Area field value is Limited

RIGHT

Format @RIGHT(*String,Num*)

String = a string value
Num = a numeric value ≥ 0

Use

@RIGHT returns the last *Num* characters of *String*. It lets you extract a specified number of characters from the right side of a string or label.

If *Num* is 0, the result is "", or an empty string. If *Num* is equal to or greater than the number of characters in *String*, the entire string is returned.

Examples

@RIGHT("Jennifer Meyer",5) = Meyer
@RIGHT("Jennifer Meyer",25) = Jennifer Meyer
@RIGHT("Jennifer ",6) = fer (including 3 trailing spaces)
@RIGHT("155",1) = 5
@RIGHT(123,1) = 3
@RIGHT(High School Complete,2) = 10, when the High School Complete field value is Grade 10

ROUND

Format @ROUND(*X,Num*)

X = a numeric value
Num = a numeric value from –15 to 15

Use

@ROUND adjusts the precision of *X* to *Num* decimal points. *Num* specifies the power of 10 to which *X* is rounded. If *Num* is positive, *X* is rounded *Num* digits to the *right* of the decimal point.

If *Num* is negative, *X* is rounded *Num*+1 digits to the *left* of the decimal point. For example, if *Num* is –3, *X* is rounded to the nearest thousand.

If *Num* is 0, *X* is rounded to an integer. If *Num* is not an integer, it is rounded off to the nearest integer.

Examples

@ROUND(12345.54321,0) = 12346
@ROUND(12345.54321,2) = 12345.54
@ROUND(12345.54321,-2) = 12300
@ROUND(Average Weight,1) = 2143.9, when the Average Weight field value is 2143.877

SECOND

Format @SECOND(*DateTimeNumber*)

DateTimeNumber = a numeric value from –36522 to 73050.99999 representing a date/time serial number

Use

@SECOND returns the second portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/serial number. Because only the decimal portion of a serial number evaluates to a time, the integer portion of the number is disregarded. The result is a number from 0 to 59.

To extract the second portion of a string that is in time format (instead of serial format), use @TIMEVALUE within the @SECOND function to translate the time into a serial number (see page 177). You can also use @TIME to enter a time value instead of a serial number (see page 177).

Examples

@SECOND(.3655445) = 23
@SECOND(.2543222) = 13
@SECOND(35) = 0
@SECOND(@NOW) = 49, if the current date and time on the computer clock running ObjectVision is March 17, 1990, 2:15:49 p.m.
@SECOND(@TIME(3,15,22)) = 22
@SECOND(@TIMEVALUE("10:08:45 am")) = 45
@SECOND(@TIMEVALUE("10:08 am")) = 0

STORE

Format @STORE(*LinkName*)

LinkName = an open Paradox, dBASE, or Btrieve database link

Use

@STORE uses the current ObjectVision data either to insert a new record into the database file; or, if the record is being viewed, to update the current ObjectVision record.

@STORE returns Yes if the current ObjectVision data is successfully written to the database file; otherwise, No.

Example

@STORE("INVEN") = Yes, when the INVEN link is open and the data is successfully transferred to the database file
@STORE("INVEN") = No, when the INVEN link is disconnected

SUM

Format @SUM(*List*)

List = two or more numeric values, up to a maximum of 14

Use

@SUM returns the total of all numeric values in *List*. *List* can be any combination of field value references, logical values, and numeric values. Text values are converted to numbers, if possible. Logical values evaluate as 1 (Yes) or 2 (No).

When more than one argument is used, arguments must be separated by commas. Any value @SUM is unable to convert to a number returns ERR (error).

Examples

@SUM(78,125) = 203
@SUM(203,417,94) = 714
@SUM(Food Expense,Lodging Expense,Travel Expense,Miscellaneous) = 714, when the Food Expense field value is 78, Lodging Expense is 125, Travel Expense is 417, and Miscellaneous is 94

TIME

Format @TIME(*Hr,Min,Sec*)

Hr = a number from 0 to 23, representing Hour
Min = a number from 0 to 59, representing Minute
Sec = a number from 0 to 59, representing Second

Use

@TIME returns the date/time serial number represented by *Hr:Min:Sec*. Each of these arguments must be within the valid ranges specified above. Any fractional portions are truncated.

Examples

@TIME(3,0,0) = 0.125 (3:00 am)
@TIME(3,30,15) = 0.14600694444 (3:30:15 am)
@TIME(18,15,59) = 0.76109953704 (6:15:59 pm)

TIMEVALUE

Format @TIMEVALUE(*TimeString*)

TimeString = a string value in any valid time format, enclosed by quotes

Use

@TIMEVALUE returns a serial time value in the range from 0 to 0.9999, calculated as the fractional portion of a 24-hour day that corresponds to the value in *TimeString*. TimeString must be a text value containing a valid time from 0:00:00 to 23:59:59.

If the value in *TimeString* is not in the correct format, or is not enclosed in quotes, an ERR value is returned.

You can enter the time string in any of the ObjectVision time display formats; the "AM" or "PM" designation is optional. There are two valid formats for *TimeString*:

- HH:MM:SS AM/PM (03:45:30 PM)
- HH:MM AM/PM (03:45 PM)

The actual time separator character used (recognized) is the one you specify in the Windows Control Panel.

Examples

@TIMEVALUE("03:30:15 AM") = 0.1460069444
@TIMEVALUE("03:00") = 0.125
@TIMEVALUE("03:45 PM") = 0.65625
@TIMEVALUE("18:15:59") = 0.76109953704
@TIMEVALUE("3.45") = ERR
@TIMEVALUE("14:00 PM") = ERR

TOP

Format @TOP(*LinkName*)

LinkName = an open ASCII, Paradox, dBASE, or Btrieve link name

Use

@TOP repositions the *LinkName* database link to the first record in the ASCII, Paradox, dBASE, or Btrieve file based on link indexing order. *LinkName* must be the name of a link previously opened with Tools | Link, or the @ASCIIOPEN, @PXOPEN, @DBOPEN, or @BTRVOPEN function.

@TOP returns Yes if successful; otherwise, No.

Examples

@TOP("ORDER"), repositions the ORDER link to the first record in the ORDER database.

TYPE

Format @TYPE(*Value*)

Value = a constant value or an expression returning a value

Use

@TYPE returns one of four codes indicating the data type of *Value*:

- **1** Numeric value
- **2** Text value
- **4** Logical value
- **16** Error value

Examples

@TYPE(State Tax) = 1, when the State Tax field value is a numeric value

@TYPE('1990 File') = 4, when the 1990 File field value is Yes
@TYPE(2.14159) = 1
@TYPE("Thomas Jefferson") = 2
@TYPE(No) = 4
@TYPE(@ERR) = 16

UPDATE

Format @UPDATE(*LinkName*)

LinkName = an open Paradox, dBASE, or Btrieve link name

Use

@UPDATE writes current ObjectVision data values to the currently viewed record of the database link. If the link specifies an order (with an index), the index is also updated.

@UPDATE will fail if there is no current link position in the database.

@UPDATE returns Yes if successful; otherwise, No.

Examples

@UPDATE("ORDER"), uses the ORDER link to revise an existing record in the open database.

UPPER

Format @UPPER(*String*)

String = a string value

Use

@UPPER returns *String* in uppercase characters. Numbers and symbols within a string are unchanged.

Examples

@UPPER(4839) = 4839
@UPPER(@LEFT("johnson",1)) = J
@UPPER("upper") = UPPER
@UPPER("Hello, world.") = HELLO, WORLD.
@UPPER("145 Bancroft Lane") = 145 BANCROFT LANE

WEEKDAY

Format @WEEKDAY(*DateTimeNumber*)

DateTimeNumber = a numeric value from –36522 to 73050 representing a date/time serial number

Use

@WEEKDAY returns the day of the week specified *DateTimeNumber*. The day is returned as an integer value from 1 (Sunday) to 7 (Saturday).

DateTimeNumber represents the number of days from December 30, 1899 to the date referenced in the expression. The earliest date available is January 1, 1800 and the latest date available is December 31, 2099.

The fractional portion of a date serial number is used by the time functions.

Any illegal dates return ERR as their value. If *DateTimeNumber* is given as text, it is converted to a serial number.

Example

@WEEKDAY(Start Date) = 7, when the Start Date field value is January 27, 1990 (a Saturday)

YEAR

Format @YEAR(*DateTimeNumber*)

DateTimeNumber = a numeric value from –36522 to 73050 representing a date/time serial number

Use

@YEAR returns the year portion of *DateTimeNumber*. The result will be a number from 1800 to 2099. If *DateTimeNumber* is given as a string, it is converted into a serial number.

Examples

@YEAR(22222) = 1960
@YEAR(End Date) = 1991, where the End Date field value is 1/27/91

C H A P T E R

9

Using the Stack Tool

You can manipulate the forms in your ObjectVision application with the Stack Tool. The Stack Tool provides a high-level, graphical method for copying, creating, deleting, and arranging forms. The following sections describe the organization of your application's form stack and explain how you use the Stack Tool.

The stack structure

The *stack order* defines the arrangement of the forms that display in the Stack Tool. The Goal form is the top form in the stack and is active when your application is first opened.

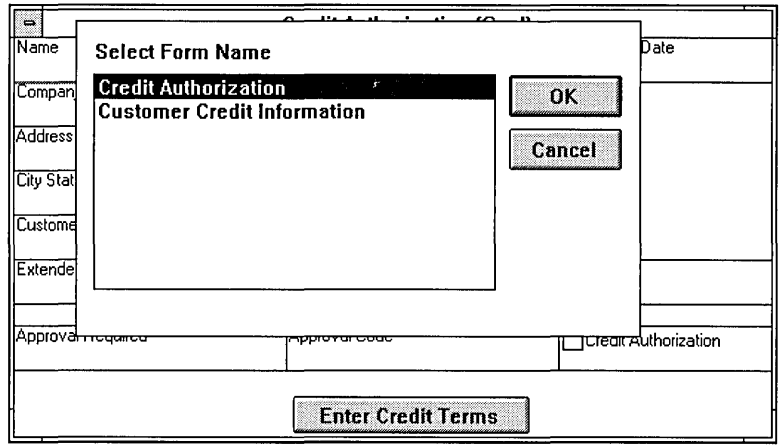
Therefore, you usually design the Goal form to contain the fields that are the goals of your application. ObjectVision omits from guided completion any forms unnecessary for completing the Goal form.

Once your application is running, the ordering of forms within the stack depends on the path taken by the user while filling out forms. As forms are selected by ObjectVision or by the user, the selected form temporarily moves to the top of the stack.

The stack order of your forms during form completion depends on which form is selected. When a user chooses the Select command from the Form menu, the Form | Select dialog box lists your application's form names in their current stack order. For

example, Figure 9.1 illustrates the initial stack order for the Credit sample application.

Figure 9.1
The Form | Select dialog box



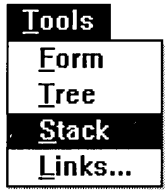
Whenever ObjectVision searches for a field, it searches each form starting at the top of the current stack and working down toward the bottom until it finds the first form containing the field. If more than one form contains the field, ObjectVision selects the form that is closest to the top of the stack.

During form completion, the form stack order is constantly changing and is not saved when your application file is saved with the File | Save or File | Save As command. Forms are stored and retrieved only according to the order defined by the Stack Tool.

Using the Stack Tool

You can use the Stack Tool for

- creating new forms
- changing the title of an existing form
- changing the order of existing forms within your application



To open the Stack Tool, choose the Stack command from the Tools menu. When you use the Stack Tool, the menu bar displays menu names unique to the Stack Tool.

The Stack Tool displays your application's forms as a list of forms. Each form displays as an icon with a title to the right of it. The top form in the stack, the Goal form, displays the word "Goal" in its icon. The selected form's title displays in reverse type.

Within the Stack Tool, ObjectVision automatically adds a vertical scroll bar. You can scroll the Stack Tool display by clicking the scroll bar or pressing *PgUp* or *PgDn*. Scroll bars appear around the Stack Tool window whenever the entire forms stack is larger than the window.

Closing the Stack Tool

When you finish using the Stack Tool, choose the Close Tool command from the Stack menu. Stack | Close Tool returns you to the previous window (form completion or form editing mode).

Creating new forms

Before adding a new form to your stack, you need to indicate where you want it by selecting a displayed form. After you choose the Objects | Form command, you can enter the title of your new form in the Objects | Form dialog box.

A new form is inserted immediately below a selected form after you choose the Objects | Form command. To insert a new form immediately above a selected form, check Insert Before in the Objects | Form dialog box.

Form titles must be unique within your application's set of forms. If you type a form name that already exists, you are prompted to type a different name. Your form title can be any text value of up to 254 characters long. But remember that long titles may not completely fit in the form's title bar.

Newly created forms have a default size of 40 characters wide and 10 characters high. After you have added a new form, you can use the Form Tool to define its objects, size, and appearance.

Editing a form stack

In addition to adding new forms to the stack, you can also cut and paste forms using the Stack Clipboard. The Stack Clipboard is a temporary holding place for a single form to be transferred within the application stack.

The Stack Clipboard is different from the Windows Clipboard and can't be used to exchange forms with other Windows applications or other ObjectVision applications.

When you use the Stack Clipboard, remember that it can only hold one cut or copied form at a time. If you want to insert a cut or copied form into the application stack, paste that form before cutting or copying again.

The following paragraphs describe the Stack Clipboard editing actions you can use.

Cut

Shift Del

You can choose Edit | Cut to delete a selected form and transfer it to the Stack Clipboard. You can use this command to remove a form from its current stack location. As a shortcut, you can press *Shift+Delete*.

Copy

Ctrl Ins

You can choose Edit | Copy to transfer a copy of a selected form to the Stack Clipboard. The selected form remains in its current stack location. As a shortcut, you can press *Ctrl+Insert*.

Paste

Shift Ins

You can choose Edit | Paste to paste a copy of the current Stack Clipboard's form to your application stack. The form is inserted immediately below the selected form.

You can enter the new form name into the Form Name dialog box that appears. If you have just cut a form from the stack, the default name will be the form's prior name.

The pasted form will have the same size and contents as the form that was copied to the clipboard. As a shortcut, press *Shift+Insert*.

Undo

Alt Backspace

You can choose Edit | Undo to reverse the previous editing action only. Once you undo that previous operation, Undo is unavailable until you perform another editing action. As a shortcut, press *Alt+Backspace.*

Reordering your forms

You can use the Edit menu commands to change the order of forms in your application stack.

To change the position of a form, you first cut it to the Stack Clipboard, select the form immediately above where you want to insert the cut form, and then paste the form from the Stack Clipboard.

Changing form titles

You can choose the Title command from the Properties menu to edit a form title you created with either the Form Tool or the Stack Tool. After you choose the Properties | Title command, the title of the selected form displays in the Properties | Title dialog box.

You can use all standard editing techniques to revise the form title, which can contain up to 254 characters. Press *Enter* when you finish editing the name.

Form titles within your application must be unique. Because ObjectVision titles are not case-sensitive, titles that differ only in capitalization are not unique and therefore are not allowed. If you enter a form title that already exists, you receive the error message "Form name already exists."

After you change a form's title, the stack display updates.

C H A P T E R

10

Linking to data files

You can create links between your application and external data files to provide field values without defining a decision tree or requiring the user to enter the data. You can also build multiple links to transfer information between a variety of sources or destinations.

For example, you can build a database link that provides field values for your application and also updates the database records with values gathered by your application. When existing data is linked to your application, you maintain data integrity and you also free the user from repetitive tasks like retyping.

You can use either @functions or the Tools | Links dialog box to create links. While both link creation methods produce the same result, sometimes one method is more appropriate than the other. The Links dialog box is easier to use and @functions are more flexible.

This chapter primarily covers creating links using the Tools | Links dialog box. Because @functions are only occasionally used for linking, they are only covered briefly in this chapter. See Chapter 8, "@Function commands," for complete information about @functions.

This chapter also defines ObjectVision links and link types and describes how to create, modify, and delete them. It covers these topics:

- what links do
- ASCII, Paradox, dBASE-compatible, Btrieve and Dynamic Data Exchange (DDE) link types
- how to create links with Tools | Links
- how to create links using @functions with buttons
- what information is required for each link type

Using links

Links perform these important functions:

- let your application get information from external data files
- let you transfer the values from your application to other applications
- let you give users a button interface for browsing and updating external files

Because ObjectVision applications link directly to external files, links create a cooperative information-processing environment where information is shared between multiple applications.

When another application changes data that an ObjectVision application is linked to, ObjectVision automatically uses the new value. Similarly, other applications can use information saved by a linked ObjectVision application.

Within ObjectVision, links are established between a specific field and an external data file. Each link can establish a connection between multiple fields and a single data file or multiple data files.

The status of links is stored in your application file along with the form and decision tree definitions. When you save an application that has active links at the time it is saved, ObjectVision automatically reconnects those same links when you open your application next time.

Link types

ObjectVision supports links to five types of external data sources: ASCII, Paradox, dBASE-compatible, Btrieve, and DDE. For each

of these types, there are @functions that manage the links between ObjectVision and the external source.

The following sections describe the supported data types and link types in detail.

ASCII files

ASCII files are simple, text-based files. ASCII files can be created by any text editor, or created by exporting database or spreadsheet data to a file.

ObjectVision links either read data from or write data to an ASCII file, but *not* both. If you want your application to read from and write to an ASCII file, you must build different links for the reading and writing operations.

ASCII file structure

ASCII files are structured as lines of text, and each line is a record containing a set of fields. For example, each time a form's values are written to an ASCII file, one record is created. Each field in a record is separated, or delimited, by a comma. When a field value contains a comma, that field must be enclosed in double quotation marks:

Doe,Jane,H.,"Apartment, duplex, or single-family"

Similarly, when a field value contains a double-quotation mark, that field must have two double-quotation marks for each occurrence:

Doe,John,R.,""""Fixer-upper"", townhouse, or single-family"

ObjectVision links create connections between the ASCII file's fields and your application's fields according to the field order in the ASCII file. ObjectVision reads or writes your application's field values to the ASCII file as specified by the link.

ASCII link options

ObjectVision automatically creates an ASCII file if you build a WRITE or APPEND link to a file that doesn't exist. You can also let ObjectVision create buttons that use link @functions to manipulate the linked ASCII file.

A WRITE link overwrites the ASCII file with the new values, and an APPEND link inserts the new values at the end of the file. If you build a READ link to an ASCII file that doesn't exist, these error messages appear:

```
Could not create ASCII link for reading
Unable to open ASCII file as specified
```

ASCII files, unlike database files, aren't structured for optimal speed so the ASCII file links are typically used for small data sets or to transfer data to or from unusual file formats. For example, an ASCII file link might be used to get constant values that customize your application for a specific group of users.

Supporting single users

ObjectVision links to ASCII files don't support multi-user activity. When ObjectVision opens a link to an ASCII file, it assumes no one else is using that file. If another application changes the file while your application is linked to it, errors will result.

Database files

A single ObjectVision link can both read from and write data to an external database file. Your ObjectVision application can use links to locate and display a specific database record, and can also write its field values to the database.

Whenever new information is read into a form from a database link, the ObjectVision application automatically recomputes any field values that are dependent on the new information.

Linking strategies

Links let you use the information in database files in a variety of ways. For example, your application might read one set of values into a form and write out different, calculated values. You can open the *Credit* sample application to see an example.

Your application might also read from and write to different records in different databases. For example, you could compare two employee's sales records for a particular quarter in the same form.

When you build an ObjectVision link, you name the link, specify an index, and define the read and write connections between the database fields and your application's fields.

Paradox, dBASE, and Btrieve

ObjectVision supports Paradox, dBASE-compatible, and Btrieve database files.

ObjectVision is 100% compatible with Paradox versions 3.0 and 3.5. ObjectVision is also compatible with dBASE III, dBASE III+, dBASE IV, Clipper, and Fox .DBF files. However, users should be

aware that ObjectVision *doesn't* support dBASE IV's .MDX index file, Clipper's .NTX index file, or Fox's .IDX file. See Appendix B, "Application limits," for more details about the database file formats ObjectVision supports.

ObjectVision automatically creates a Paradox .DB table, a dBASE-compatible .DBF data file, or a Btrieve table if it does not exist.

You can also let ObjectVision create buttons that use link @functions to manipulate the linked Paradox, Btrieve, or dBASE-compatible file. For existing Btrieve tables, you *must* create a data dictionary using the separate application XQL from Novell.

File conventions

ObjectVision links observe the data-handling conventions of the database file type. For example, when an ObjectVision link deletes a record from a Paradox table the record is physically removed. By contrast, when an ObjectVision link deletes a record from a dBase table the record is flagged as deleted until the table is packed. The difference is that dBase provides for "soft delete" and Paradox doesn't.

Data conversion

ObjectVision automatically performs data conversions on linked database field values. For example, when a database field containing a date is connected to your application's text field, ObjectVision reads the date value, converts it, and displays it as a string value.

Using indexes

Using a database index file increases the speed of your applications by directly locating specific database records. ObjectVision supports primary and secondary Paradox indexes, and dBASE-compatible .NDX index files. When a Paradox .DB file or a dBASE .DBF file is created automatically by ObjectVision, the first field in the form becomes the default primary index.

An index is made up of one or more fields and is used to reposition the link to a desired database record. Whenever a value in the indexed fields changes, ObjectVision automatically repositions the link to the matching record in the database.

If the linked dBase database file has multiple indexes, you must create a link for each index file if you want ObjectVision to automatically update all of the indexes. Otherwise, the indexes must be updated manually in a separate database application.

Multi-user support

ObjectVision's Paradox, dBASE-compatible and Btrieve links support multiple user activity for both reading and writing to database files.

ObjectVision only supports Clipper's file locking for dBASE-compatible files. DBASE's network file locking is not supported because it opens files exclusively for the dBASE application opening the file.

A user can always open a linked file for read access, unless the file itself is locked. When a linked database file is changed by an external application, that modified data displays the next time ObjectVision reads that record.

Before ObjectVision changes a record in the database, it reads the existing database to determine whether another application revised that record while your ObjectVision application was modifying it.

When a record with the same index key exists and it has different values than when your ObjectVision application originally read it, a dialog box appears that alerts users. A user must either confirm or cancel overwriting these unseen changes with their current field values.

(DDE) Windows applications

ObjectVision DDE links use the Dynamic Data Exchange (DDE) protocol that works with any Windows product supporting DDE. A DDE link is a read-only link that connects named data in the other application with your ObjectVision fields.

DDE links transfer data instantly from one Windows application to another, so they are an effective way for applications to share constantly changing information.

Unlike ASCII and database links, DDE links require both a data file and the application that created that file. If the other application is not loaded in memory, ObjectVision attempts to start the application in order to connect the link.

The Windows application you are linking to *must* be in your PATH definition.

Reading ObjectVision values

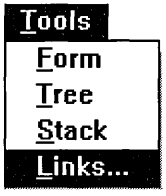
After the link is established to the other application, it sends a message to your ObjectVision application when its linked data changes and it also updates your application's field values. ObjectVision automatically recalculates any field values that are dependent on DDE link values.

Exporting ObjectVision values

To export information from your ObjectVision application to another application, you must create a read-only link to your ObjectVision application from within the other application.

You can use DDE links to break up a large ObjectVision application into separate, smaller applications. Using DDE, another application can request your application to send data back. When you don't want another application to receive the values, you can set your ObjectVision application to ignore DDE requests.

Creating links



You can create and manage external links in either of two ways:

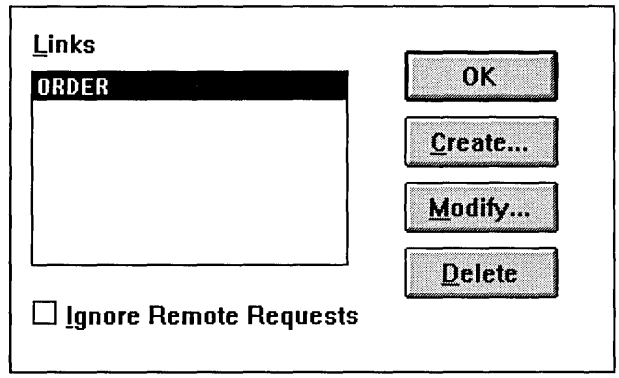
- By choosing Tools | Links. The Tool | Links dialog box lets you select the type of link you want and what link action you want to carry out.
- By including @functions in decision trees. ObjectVision provides functions for each link type and link operation.

Using Tools | Links

You can use Tools | Links to create, modify, and delete external links. The Tools | Links dialog box displays the link names (if any) you already defined, and the operations you can perform on the links.

Figure 10.1 shows the Tools | Links dialog box.

Figure 10.1
Tools | Links dialog box



Each ObjectVision link you build must be given a unique link name within the ObjectVision application.

You choose the link operations you want from the right side of the Tools | Links dialog box. These operations apply to the currently selected link in the Links list. When a command button is unavailable for the selected link, its name is dimmed.

OK When you choose OK, the Tools | Links dialog box closes.

Create When you choose Create to define a new link, the Tools | Links | Link Type dialog box appears as shown in Figure 10.2. After you select the link type you want, choose OK to carry out the action; the link type-specific dialog box appears, where you specify your link information.

Modify When you choose Modify to change the selected link's attributes, the link-specific dialog box appears for that link. The current link specifications display in the dialog box and you can change any of the settings.

New settings you select take place immediately. If you decide not to save the new settings, choose Cancel to close the dialog box and discard the changes. ObjectVision notifies you if it can't establish the modified link.

Delete

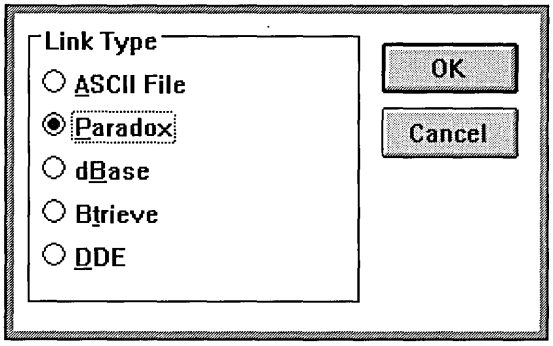
You can choose Delete to disconnect and remove the selected link. This action is immediate and you can't undo it.

Ignore Remote Requests

You check Ignore Remote Requests to prohibit other Windows applications from reading your application's data. This option applies only to external DDE requests from other applications. This does not prohibit your application from requesting data from other applications through DDE.

To create a new link, you choose Tools | Links and then choose Create. The Tools | Links | Link Type dialog box appears as shown in Figure 10.2, where you select the type of link you want to build. Your next steps depend on the type of link type you select.

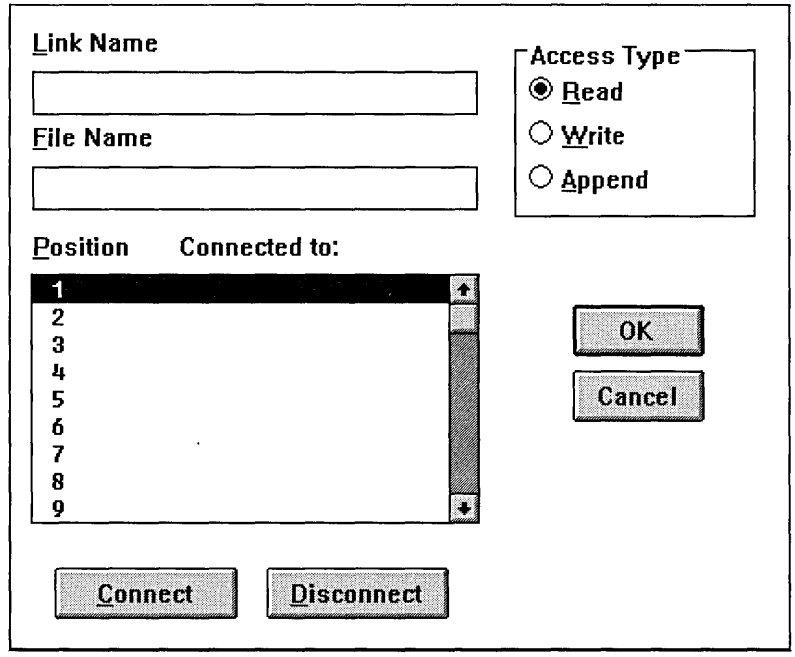
Figure 10.2
Link Type dialog box



Linking to ASCII files

You can establish links to ASCII files with Tools | Links or with @ASCIIOPEN. The Link Type | ASCII dialog box shown in Figure 10.3 is the simplest way to create an ASCII link. This section explains how to build an ObjectVision link to an ASCII file using the Link Type | ASCII dialog box.

Figure 10.3
Link Type I ASCII dialog box



Link Name

First, you type a unique link name, which can use any ANSI characters and be up to 63 characters long. A link name identifies a directional link between an ASCII file and your application.

The name you choose might help you remember the purpose of the link. For example, if you are creating a link to append records to an existing file, you might want to name the link *ADD RECORDS*.

When you create a link, avoid using double-quotation marks in the link name you supply. When a link name is used in an expression or as an argument, it must be enclosed in double-quotation marks. If a link name contains a double-quotation mark, the double-quotation marks must be typed twice.

File Name

Next, type the name of the ASCII file you want to link to. The file name must be a standard DOS file name, with a maximum of 8

characters for the name and a maximum of 3 characters for the file extension.

If the file you are linking to is not in the same directory as your ObjectVision application, you must include the path name. The full path name can be up to 63 characters long.

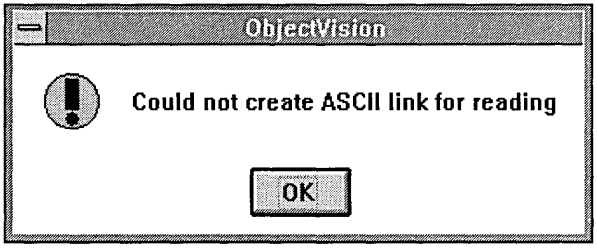
Option

Option defines the type of data transfer you want to take place between an ObjectVision application and its linked ASCII file. An ASCII file can be opened for either reading, writing, or appending data.

READ This option creates a read-only link; the file can't be modified through a READ link and ObjectVision doesn't create an ASCII file when one doesn't exist. When an ASCII file is opened, the values in its first record (the first line of the file) display in the associated ObjectVision fields. Records on subsequent lines within the ASCII file are read sequentially.

Unless a full path name is provided, ObjectVision looks for the ASCII file in the current directory; if it is not found ObjectVision searches the directories specified in the path environment variable. If the file name you entered doesn't exist, the dialog box shown in Figure 10.4 appears.

Figure 10.4
READ link error dialog box



After you choose OK, this error message appears:

```
Unable to open ASCII file as specified
```

Choose OK to continue.

WRITE When an ASCII file is opened for writing, that file is overwritten each time the application is used. Each session that opens the link replaces the old file with a new file. If the file doesn't exist, it is created automatically.

The associated ObjectVision field values are not written to a new line in the ASCII file until the @INSERT function is carried out. The @INSERT function can be carried out repeatedly to save multiple records to the same ASCII file.

If the file already exists, ObjectVision prompts you to confirm that you want to create a new file using the same name as an existing file. If you choose Yes, the new data is written over the existing data and the old file is lost. If you choose No, the link is not connected and the data is not written to the file.

APPEND When an ASCII file is opened for appending, field values are written to that file each time @INSERT is carried out. The new record is written after the last record in the ASCII file.

Any existing records in the ASCII file remain unchanged. If the named ASCII file does not exist, it is created automatically.

Position

The Position column in the Link Type | ASCII dialog box displays numbers corresponding to the order of field positions in the ASCII file. For example, the first field in the ASCII file is labeled 1 in the Position column. You might want to print a list of the ASCII field names for reference while connecting them to fields.

Connect To connect a field in the ASCII file to an ObjectVision field, you first select a number in the Position column, then press Connect. The Field Name dialog box lists all fields in the ObjectVision application. Select the field you want to link to the selected field position in the ASCII file.

You can continue linking fields until you connect all the fields you want. When you choose OK, ObjectVision links the field to the appropriate position in the ASCII file; ObjectVision doesn't read the ASCII file's data until the link is fully created and it is a read link.

Disconnect Any field in the ASCII file that doesn't have a corresponding field in the ObjectVision application will be skipped. If you want to remove an existing connection to an ASCII field, press Disconnect.

Choose OK to save the link when you finish. If there is an error in the link definition, an error message appears and you can revise the link settings. When you choose OK and the new link definition is correct, this message appears:

```
Automatically add appropriate buttons for new link named
LinkName?
```

where *LinkName* is the link name you typed.

Choose OK if you want ObjectVision to put buttons for the new link on your form. Next is added for READ links, and Clear and Insert are added for WRITE and APPEND links.

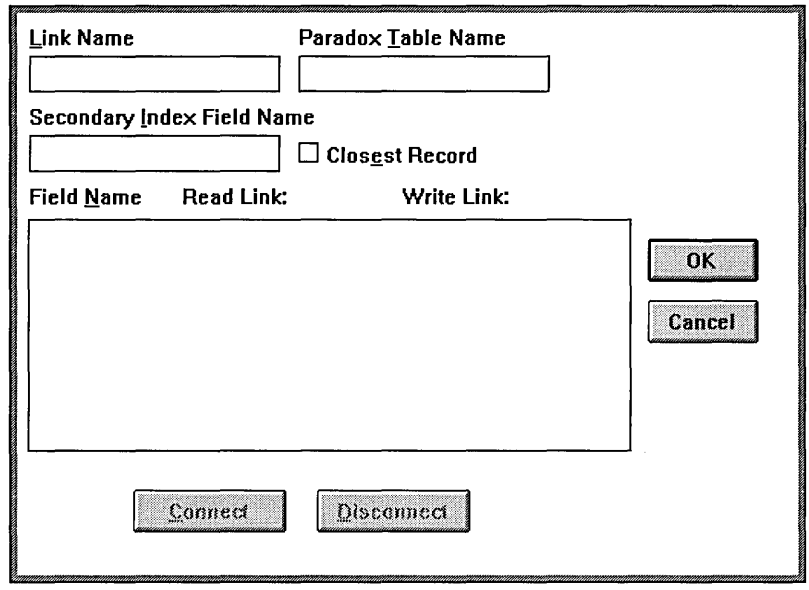
Clear uses the @CLEAR function in its decision tree to remove user-entered values from the form and it also prepares the ASCII file for the next record entry. Enter uses the @INSERT function to write the values to the ASCII file. You can choose No if you don't want these default buttons put on your form.

After you choose Yes or No, the Tools | Links dialog box appears so you can continue working with links.

Linking to Paradox files

You can build links to Paradox tables using Tools | Links or with @PXOPEN. Figure 10.5 shows the Link Type | Paradox dialog box that appears when you create a new Paradox link. The steps are similar to those used to create other ObjectVision links.

Figure 10.5
Link Type I Paradox dialog box



Link Name

First, you type a unique link name, which can use any ANSI characters and be up to 63 characters long. A link name identifies a bi-directional link between a Paradox table and your application.

The name you choose might help you remember the purpose of the link. For example, if you are creating a link to store values in an invoice table, you might want to name the link *STORE INVOICE*.

When you create a link, avoid using double-quotation marks in the link name you supply. When a link name is used in an expression or as an argument, it must be enclosed in double-quotation marks. If a link name contains a double-quotation mark, the double-quotation marks must be typed twice.

File Name

Next, type the name of the Paradox table you want to link to. The name must be a standard DOS file name, with a maximum of 8 characters for the name. Paradox tables have a .DB extension after the file name, but it isn't necessary for you to type the extension.

If the file you are linking to is not in the same directory as your ObjectVision application, you must include the path name. The full path name can be up to 63 characters long.

After you select another field in the dialog box, ObjectVision reads the Paradox file, if it exists. When the Paradox file exists, the list of field names displays in the Name column.

If the Paradox table does not exist, a message appears asking if you want to create a Paradox table automatically. If you choose Yes, each field in the new table is created based on the size and type of the fields in your *active form* only. A primary index field is also created along with the table, based on the first field in your form.

Once the table is created, it cannot be changed from within ObjectVision; to restructure the table you must use Paradox.

Primary index

ObjectVision automatically uses the primary index for a linked Paradox table unless a secondary index is indicated. The primary index has a .PX extension and must be in the same directory as the .DB table.

All indexed fields in a Paradox table must be connected to an ObjectVision field. Indexed fields only need to be connected to a Read Link. To use a primary index, connect the first *n* fields of the Paradox table to ObjectVision fields, where *n* is the number of fields in the primary key.

Secondary Index Field Name

Paradox supports secondary indexes to provide different access methods and sort orders. To use a secondary index, type the name of the single Paradox field that you want to use to locate records or control record sequence.

Closest Record

When you want to let users search for existing records, you can check Closest Record. Closest Record is typically used when users are changing existing data, not when entering new records.

Closest Record lets a user search Paradox tables using either the primary or secondary index field, by typing a value into the

connected index field and pressing *Enter*. Closest Record locates the closest approximation to the entered value in the database (greater than or equal to).

For example, if a user is looking for a record with the name Smith during form completion, they can type *Sm* into the indexed field, then press *Enter*, and the first record beginning with *Sm* displays. If no records begin with *Sm*, then the next closest record displays.

When a Paradox table has multiple index fields, those fields must be put on the ObjectVision form and connected using the Tools | Link dialog box. Additionally, Closest Record only works properly when values for the ObjectVision fields are entered in the order of the index fields in the Paradox table.

For example, when both Last Name and First Name are indexed fields, the closest record is found when values are entered into only one of the fields. ObjectVision uses a blank character when one field is left blank, and the search returns the first value in the table for that field (because blanks come before characters in the sort order).

Name

To establish a link between a Paradox table field and an ObjectVision field, first select a Paradox field name.

Connect

After you choose Connect, the Read Field dialog box lists the existing fields in your ObjectVision application. A field you select in this dialog box is the field that connects to the highlighted Paradox field.

You can also create a new ObjectVision field by selecting <Add New Field> and entering a name in the New Field Name dialog box. After you choose OK, the Write Field dialog box appears.

Next, a list of fields appears for the Write Field Name. Select the ObjectVision field which contains the value you want to write to the Paradox file. Again, you can also create a new ObjectVision field by selecting <Add New Field> and entering a name in the New Field Name dialog box.

You can continue connecting the fields until all links are defined. You can leave any Read Field or Write Field disconnected.

Disconnect

If you want to change the ObjectVision field that a Paradox field is connected to, select the field, then press Disconnect. This removes the connection for that field.

Typically, the same ObjectVision field is used to connect both the Read Field and the Write Field to a Paradox field; but it isn't required. For example, if you need to read a Paradox field without modifying it, you can select <Not Connected> in the Write Field dialog box. Or, if you need to write a value to the Paradox field without reading the current value, you can select <Not Connected> in the Read Field dialog box.

OK

Choose OK to save the link when you finish defining your Paradox link.

If you enter a table name that doesn't exist, this message appears:

```
Unable to open Paradox table. Create a new table named
FileName?
```

where *FileName* is the name you specified. If you choose Yes, the table is created automatically; if you choose No, this message appears:

```
Table not found!
```

If there is an error in the link definition you can revise the link settings. When you choose OK in the Link Type | Paradox dialog box and the new link definition is correct, this message appears:

```
Automatically add appropriate buttons for new link named
LinkName?
```

where *LinkName* is the link name you typed.

Choose OK if you want ObjectVision to put Previous, Clear, Enter, Next, and Delete buttons for the new link on your form as shown in Figure 10.6.

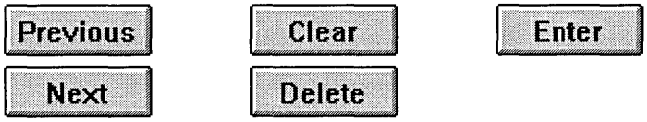


Figure 10.6
Buttons added by ObjectVision

Previous uses the @PREVIOUS function in its decision tree to move to the previous record in the Paradox file. Next uses the @NEXT function to move to the next record in the Paradox file.

Clear uses the @CLEAR function to remove user-entered values from the form, and it also prepares the database for a new record entry. Delete uses the @DELETE function to erase records in the Paradox file.

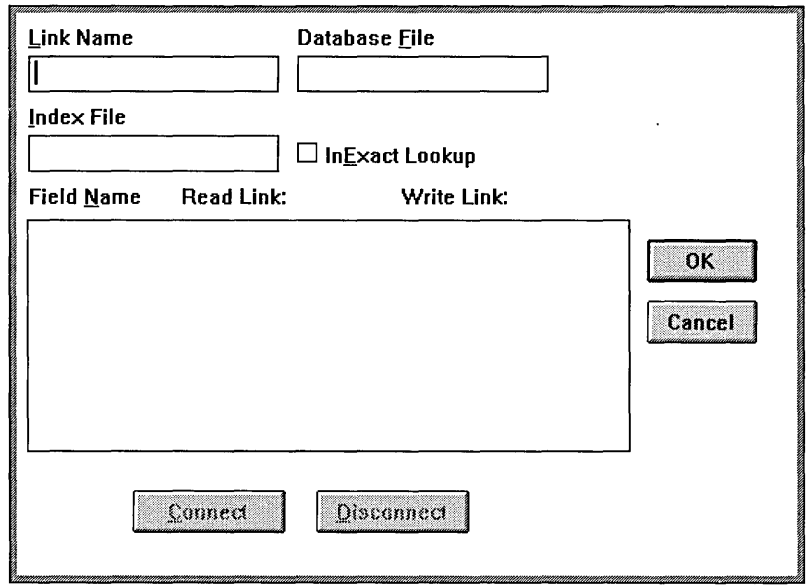
Enter uses the @STORE function to write the values to the Paradox file.

You can choose No if you don't want these default buttons put on your form. After you choose Yes or No, the Tools | Links dialog box appears so you can continue working with links.

Linking to dBASE-compatible files

You can build links to dBASE-compatible files with Tools | Links or with @DBOPEN. Figure 10.7 shows the Tools | Links | Link Type | dBase dialog box that appears when you create a new dBASE link. The steps are similar to those used to create other ObjectVision links.

Figure 10.7
Link Type | dBase dialog box



Link Name

First, you type a unique link name, which can use any ANSI characters and be up to 63 characters long. A link name identifies a link between a dBASE-compatible file and your application.

The name you choose might help you remember the purpose of the link. For example, if you are creating a link to constant values in a lookup table, you might want to name the link *LOOKUP VALUES*.

When you create a link, avoid using double-quotation marks in the link name you specify. When a link name is used in an expression or as an argument, it must be enclosed in double-quotation marks. If a link name contains a double-quotation mark, the double-quotation marks must be typed twice.

File Name

Next, type the name of the dBASE-compatible data file you want to link to. The file name must be a standard DOS file name, with a maximum of 8 characters for the name. dBASE files have a .DBF extension after the file name, but is isn't necessary for you to type the extension.

If the file you are linking to is not in the same directory as your ObjectVision application, you must include the path name. The full path name can be up to 63 characters long.

Index files

If you want to be able to directly search for records in the dBASE-compatible file, you must type the name of the dBASE index file. An index file controls the sort order of the data file. A dBase index file has the extension .NDX, but it isn't necessary to type the extension.

ObjectVision links support the same number of index fields that dBASE does. All indexed fields in a dBASE-compatible file must be connected to an ObjectVision field.

The index file is optional for sequential access, where a user simply displays one record after another in the order the records were written to the database.

InExact Lookup

When you want to let users search for existing records, you can check InExact Lookup. InExact Lookup is typically used when users are changing existing data, not when entering new records.

InExact Lookup lets a user search dBASE-compatible files using the linked index fields, by typing a value into the connected index field and pressing *Enter*. InExact Lookup locates the first record matching the entered value.

For example, if a user is looking for a record with the name Smith during form completion, they can type *Sm* into the indexed field, then press *Enter*, and the first record beginning with *Sm* displays. However, if no records begin with *Sm*, then no record displays.

When multiple fields in the dBASE file are indexed, those fields must be put on the ObjectVision form and connected using the Links | dBase dialog box. Additionally, InExact Match only works properly when values for the ObjectVision fields are entered in the same order as the index fields in the dBASE file.

For example, when both City and State are indexed fields, an inexact match is found if a value is entered in City, or in both City and State. When a user enters a value into multiple index fields in the proper order, ObjectVision searches for the Inexact Match using all of the specified index field values. If an indexed field is left blank during a search, the blank character is used by Object-Vision and the first match for the blank field is located (because the blank character comes before the character *a* in the sort order).

Name

To establish a link between a field in a dBASE-compatible file and an ObjectVision field, first select a dBASE field name, then choose Connect.

Connect

After you choose Connect, the Read Field dialog box lists the existing fields in your ObjectVision application. A field you select in this dialog box is the field that connects to the highlighted dBASE field.

You can also create a new ObjectVision field by selecting <Add New Field> and entering a name in the New Field Name dialog box. After you choose OK, the Write Field dialog box appears.

Next, a list of fields appears for the Write Field Name. Select the ObjectVision field which contains the value you want to write to the dBASE file. Again, you can also create a new ObjectVision field by selecting <Add New Field> and entering a name in the New Field Name dialog box.

You can continue connecting the fields until all links are defined. You can leave any Read Field or Write Field disconnected.

Disconnect

If you want to change the ObjectVision field that a dBASE field is connected to, select the field, then press Disconnect. This removes the connection for that field.

Typically, the same ObjectVision field is used to connect both the Read Field and the Write Field to a dBASE field; but it isn't required. For example, if you need to read a dBASE field without modifying it, you can select <Not Connected> in the Write Field dialog box. Or, if you need to write a value to the Paradox field without reading the current value, you can select <Not Connected> in the Read Field dialog box.

OK

Choose OK to save the link when you finish defining your dBASE link.

If there is an error in the link definition, an error message appears and you can revise the link settings. When you choose OK and the new link definition is correct, this message appears:

```
Automatically add appropriate buttons for new link named
LinkName?
```

where *LinkName* is the link name you typed.

Choose OK if you want ObjectVision to put Previous, Clear, Enter, Next, and Delete buttons for the new link on your form.

Previous uses the @PREVIOUS function in its decision tree to move to the previous record in the dBASE file. Next uses the @NEXT function to move to the next record in the dBASE file.

Clear uses the @CLEAR function to remove user-entered values from the form. Delete uses the @DELETE function to mark records in the dBASE file for deletion. ObjectVision can't read a record that is marked for deletion, but the record remains in the database

until it is erased from within the dBASE application using the Pack command.

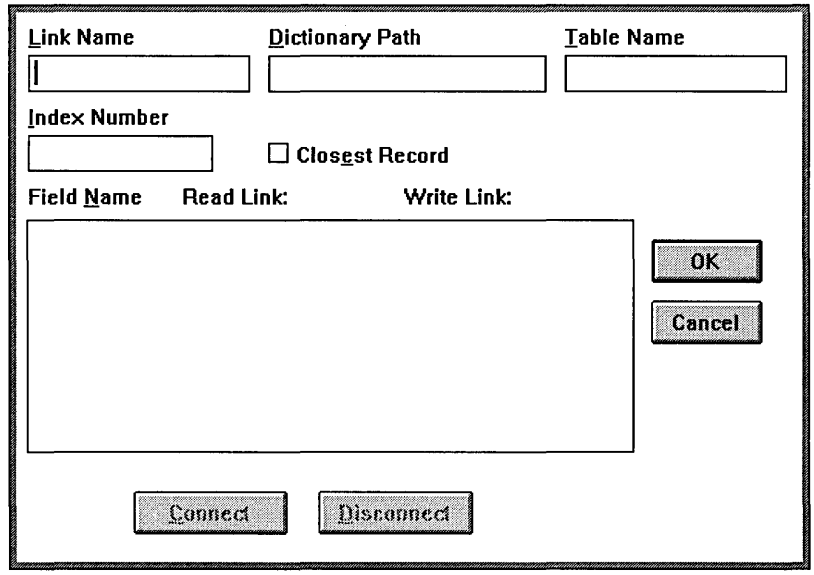
Enter uses the @STORE function to write the values to the dBASE file.

You can choose No if you don't want these default buttons put on your form. After you choose Yes or No, the Tools | Links dialog box appears so you can continue working with links.

Linking to Btrieve files

You can build links to Btrieve files with Tools | Links or with @BTRVOPEN. Figure 10.8 shows the Link Type | Btrieve dialog box that appears when you create a new Btrieve link. The steps are similar to those used to create other ObjectVision database links.

Figure 10.8
Link Type | Btrieve dialog box



For linking to existing Btrieve tables, you need *both* Novell's Btrieve and XQL applications in order to create your database tables and the data dictionary files for them. The XQL application must be used separately to create a set of three files that contain the Btrieve field definitions. The files are named FIELD.DDF, FILE.DDF, and INDEX.DDF.

If you use ObjectVision to create your Btrieve database files, you don't need either Novell's Btrieve or XQL applications. The tables and the data dictionary files are automatically created using the information you supply in the Btrieve Link dialog box.

However, you will need Btrieve and XQL if you want to modify the file structure or moves the files to another directory.

Link Name

First, you type a unique link name, which can use any ANSI characters and be up to 63 characters long. A link name identifies a link between a Btrieve file and your application.

The name you choose might help you remember the purpose of the link.

Avoid double-quotation marks in link names; it will make writing expressions easier. When a link name is used in an expression or as an argument, it must be enclosed in double-quotation marks. If a link name contains a double-quotation mark, the double-quotation marks must be typed twice.

Dictionary path

Next, type the directory path for the directory containing the set of three .DDF files created by XQL (FIELD.DDF, FILE.DDF, and INDEX.DDF) that contain the field definitions for the table you want to link to.

The file name must be a standard DOS file name, with a maximum of 8 characters for the name. The full path name can be up to 63 characters long.

For an existing Btrieve file you *must* create a dictionary using Novell's XQL or other Novell applications.

Table Name

Next, type the name of the Btrieve table you want to link to. The name must be a standard XQL table name, with a maximum of 20 characters for the name. The table's field names and field types information must be contained in the Dictionary.

Index Number

Type the index number which specifies the pre-defined indexing scheme to use. This is the index number you selected when you created the index in Btrieve (an existing index number from 0 to 23).

Closest Record

When you want to let users search for existing records, you can check Closest Record. Closest Record is typically used when users are changing existing data, not when entering new records.

Closest Record lets a user search Btrieve files using the linked index fields, by typing a value into the connected index field and pressing *Enter*. Closest Record locates the record closest to the index value, just like in an ObjectVision Paradox link.

Field Name

To establish a link between a field in a Btrieve table and an ObjectVision field, first select a Btrieve field name, then choose Connect.

Connect

After you choose Connect, the Read Field dialog box lists the existing fields in your ObjectVision application. A field you select in this dialog box is the field that connects to the highlighted Btrieve field.

You can also create a new ObjectVision field by selecting <Add New Field> and entering a name in the New Field Name dialog box. After you choose OK, the Write Field dialog box appears.

Next, a list of fields appears for the Write Field Name. Select the ObjectVision field which contains the value you want to write to the Btrieve file. Again, you can also create a new ObjectVision field by selecting <Add New Field> and entering a name in the New Field Name dialog box.

You can continue connecting the fields until all links are defined. You can leave any Read Field or Write Field disconnected.

Disconnect

If you want to change the ObjectVision field that a Btrieve field is connected to, select the field, then press Disconnect. This removes the connection for that field.

Typically, the same ObjectVision field is used to connect both the Read Field and the Write Field to a Btrieve field; but it isn't required.

OK

Choose OK to save the link when you finish defining your Btrieve link.

If there is an error in the link definition, an error message appears and you can revise the link settings. When you choose OK and the new link definition is correct, this message appears:

```
Automatically add appropriate buttons for new link named
LinkName?
```

where *LinkName* is the link name you typed.

Choose OK if you want ObjectVision to put Previous, Clear, Enter, Next, and Delete buttons for the new link on your form.

Previous uses the @PREVIOUS function in its decision tree to move to the previous record in the Btrieve file. Next uses the @NEXT function to move to the next record in the Btrieve file.

Clear uses the @CLEAR function to remove user-entered values from the form, and it also prepares the database for a new record entry. Delete uses the @DELETE function to erase records in the Btrieve file.

Enter uses the @STORE function to write the values to the Btrieve file.

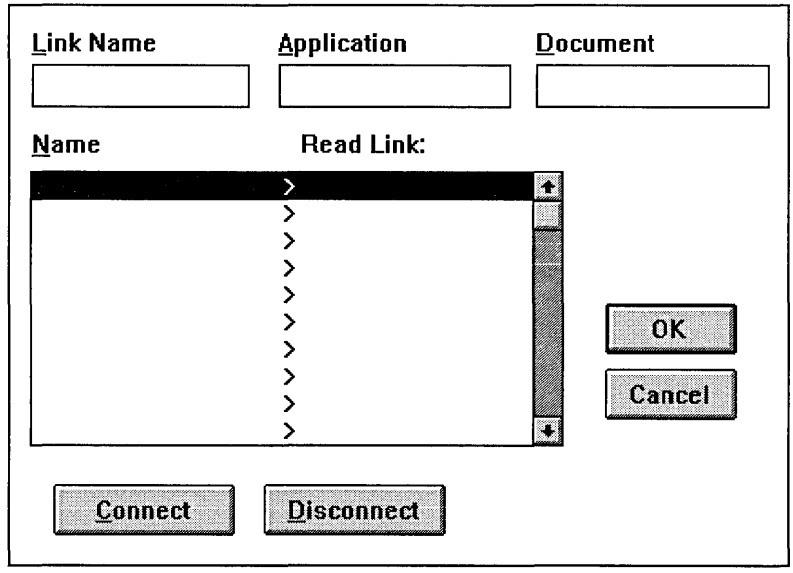
You can choose No if you don't want these default buttons put on your form. After you choose Yes or No, the Tools | Links dialog box appears so you can continue working with links.

After you choose Yes or No, the Tools | Links dialog box displays so you can continue working with links.

Linking through DDE

You can use either Tools | Links or @DDEOPEN to establish DDE links to other Windows applications. Figure 10.9 shows the dialog box that appears after you choose Create in the Tools | Links dialog box. The steps you use to create a DDE link are similar to those used to create other ObjectVision links.

Figure 10.9
Link Type | DDE dialog box



Link Name

First, you type a unique link name, which can use any ANSI characters and be up to 63 characters long. A link name identifies the read-only link between your application and another Windows application.

The name you choose might help you remember the purpose of the link. For example, if you are creating a link to read spreadsheet totals in a quarterly report, you might want to name the link *QTR TOTALS*.

Avoid double-quotation marks in link names; it will make writing expressions easier. When a link name is used in an expression or as an argument, it must be enclosed in double-quotation marks. If a link name contains a double-quotation mark, the double-quotation marks must be typed twice.

Application

You type the name of the Windows application you are building a link to. The application name (including path name) can be up to 63 characters long.

The application name is typically the .EXE file name. For example, to create a DDE link to another ObjectVision application, you must type the application name `VISION`; to Excel, you must type `Excel`.

If the application you want to link to is not already loaded in memory, ObjectVision attempts to load and run the application.

Document

Next, you can type the name and extension of the document file you want your application to read. For example, to link to another ObjectVision application, you type that file name; for Excel, you type the worksheet file name.

The file name must be a standard DOS file name, with a maximum of 8 characters for the name and 3 characters for the extension. For example, ObjectVision files have a .OVD extension after the file name; Excel worksheet files have an .XLS extension and macro worksheets have an .XLM extension.

If the file you are linking to is not in the same directory as your ObjectVision application, you must include the path name. The full path name can be up to 63 characters long.

Name

To establish a link between a DDE document's named data area and an ObjectVision field, first select the first row in Name, then choose Connect.

Connect

After you choose Connect, a dialog box prompts you to enter a Remote Name for the external data file's field.

The Remote Name varies depending on the DDE application. For example, in ObjectVision, the Remote Names are the application's field names; in Excel, they are cell names defined with the Formula Define Name command.

After you choose OK to enter the Remote Name, the Read Field dialog box lists the existing fields in your ObjectVision application. A field you select in this dialog box is the field that connects to the highlighted DDE remote name.

You can also create a new ObjectVision field by selecting <Add New Field> and entering a name in the New Field Name dialog box.

OK When you finish connecting remote names to your ObjectVision fields, choose OK. ObjectVision immediately attempts to establish the DDE link with the other application. If the other application is not loaded in memory, ObjectVision attempts to start the application and open the document.

If the DDE link can't be opened, an error message appears and you can revise the link settings. You can avoid path errors by including Windows applications' directories in your PATH statement.

When you finish revising your link definition and all errors are corrected, choose OK. The name of the new link you defined displays in the DDE Link dialog box.

Linking with @functions

Each ObjectVision link type supports specific @functions, as shown in Table 10.1. You can type or paste link functions into decision trees just like other functions.

For more information about @functions and their arguments and usage, see Chapter 8, "@Function commands."

Table 10.1
ObjectVision link functions by data file type

ASCII	PARADOX	dBASE	Btrieve	DDE
@ASCIIOPEN	@PXOPEN	@DBOPEN	@BTRVOPEN	@DDEOPEN
@CLOSE	@CLOSE	@CLOSE	@CLOSE	@CLOSE
@TOP	@TOP	@TOP	@TOP	
	@BOTTOM	@BOTTOM	@BOTTOM	
	@PREVIOUS†	@PREVIOUS†	@PREVIOUS†	
@NEXT[R]	@NEXT†	@NEXT†	@NEXT†	
@CLEAR[W]	@CLEAR†	@CLEAR†	@CLEAR†	
	@DELETE†	@DELETE†	@DELETE†	
	@UPDATE	@UPDATE	@UPDATE	
@INSERT[W]	@INSERT	@INSERT	@INSERT	
	@STORE†	@STORE†	@STORE†	

†The default buttons ObjectVision creates for a database link.

[R]An ASCII READ-link only default button.

[W]An ASCII WRITE- or APPEND-link default button.

@Functions and buttons

You typically use both the Tools | Links dialog box and buttons that have @functions in their decision tree. However, it is possible to use only @functions to establish your links. @Functions give you more flexibility, but also require more steps to create a link.

Using @functions, links can be connected or disconnected at the press of a button. An ObjectVision application can have multiple links, which lets your applications transfer data from a variety of different sources.

For example, your application might read weekly sales figures from four regional offices. The same form could be used to read from or write to each of the four separate databases. This complex data manipulation is easily understood by users, who simply press labeled buttons.

Listing @function arguments

Every link function requires you to type the appropriate arguments for that function. When you use the Tools | Links dialog box, the same information is required, but you provide it using the dialog box.

You type or paste link functions into decision trees just like other functions. Typically, a link @function is used in a button. For example, the following steps create a button that repositions a link to the next record of a file already linked with Tools | Links.

Creating a link button

First, choose Tools | Form to move to the Form Tool, then choose Objects | Field to create a new field. After you name this new field Next, put the field on your form.

While Next is still highlighted, choose Properties | Field Type, select Button, and then choose OK. Choose Tools | Tree to display the decision tree for Next; it is currently empty. Choose Objects | Conclusion to define a conclusion expression for Next. Type this expression in the Conclusion dialog box:

```
@NEXT("Link Name")
```

where *Link Name* is the name of the existing link. Press *Enter*.

You can test your Next button by choosing Form | Close Tool to return to form completion mode. Choose Next to display the next record in the linked file. A message appears that lets you know when you reach the end of the file.

C H A P T E R

11

Distributing your applications

Once you have developed an application with ObjectVision, you can distribute it to other ObjectVision users. This chapter explains how to distribute an application using ObjectVision Runtime.

ObjectVision Runtime

ObjectVision Runtime provides the same user interface as the development version for form completion. ObjectVision Runtime provides all these form completion capabilities:

- guided completion
- opening and saving application files
- printing, moving, and resizing forms
- data entry for all field types
- Clipboard editing operations
- supporting users with context-sensitive and field-specific Help
- viewing decision tree logic
- selecting forms and fields
- minimal recalculation for data changes
- scrolling of large forms
- changing system colors

Completing forms only

ObjectVision Runtime uses the same menu structure as the development version, except that the Tools menu is not available. Users of ObjectVision Runtime cannot modify your applications.

Protecting calculated fields

You can use ObjectVision's protection features to prohibit users from overriding calculated field values. You can use the Form Tool to format fields as *protected*, which prohibits users from changing the value of a calculated field.

Another formatting option prohibits users from viewing the decision tree logic associated with a field. You can use this option if you do not want to disclose the decision process used within an application.

Because users of ObjectVision Runtime are unable to use the Form Tool to change these formatting options, they cannot circumvent any of your application's protection features.

ObjectVision Runtime requirements

The system requirements for ObjectVision Runtime are like those of the development version. Because the tools and associated help text are not included, the run-time executable file is significantly smaller than the development version.

The requirements for an IBM Personal Computer, Personal System/2, or 100% compatible personal computer:

- 80286, 80386, or 80486 central processing unit
- DOS Version 3.2 or later
- Microsoft Windows 3.0
- At least 640K of system memory
- At least approximately 1MB of free hard disk storage
- 1.2 MB (5.25 inch) or 1.44 MB (3.5 inch) disk drive
- High-resolution display and adapter (EGA, VGA, or Monochrome Graphics)

The following system components are optional but highly recommended:

- 512K or more of extended memory
- Mouse or other pointing device
- Printer with graphics capability

Distributing your applications

To distribute a run-time version of your application, simply include your application file and any associated data or graphics files with ObjectVision Runtime.

A user can read from and write to multiple files with ObjectVision Runtime. Users who already use a registered copy of ObjectVision Runtime can immediately use your application files.

In addition to your ObjectVision application file, you must distribute any files required by the application. Your application might require any of these related files:

- ASCII files referenced through external links
- Database and index files referenced through external links
- Other Windows applications or data files referenced through DDE links
- Bitmap files or Windows metafiles for graphics contained on forms

PART 2

Appendixes and Glossary

A P P E N D I X

A

Keyboard and mouse operations

This appendix lists the keys, key combinations, and mouse techniques you can use to perform ObjectVision operations. Keys and mouse techniques are listed in two main categories:

- for running applications
- for using ObjectVision tools

Mouse techniques you can use in ObjectVision immediately follow the keys listed in each category. This mouse symbol indicates that the text next to it begins the list of mouse techniques.

Choosing menu commands

When a menu command has a shortcut key or key combination, you can press the specified key or keys to directly carry out the action. The shortcut key or key combination, if it exists, follows the command name on the menu. For example, on the ObjectVision menu, *Alt+F4* follows the Close command.

You must choose other menu commands by selecting the menu you want, opening the menu, and choosing the command you want.

Alt Selects the menu bar. By default, the first menu name is highlighted.

← or →

or

After you have selected the menu bar, you can press ← or → to highlight the menu name you want.

Enter

or

↓

Opens the selected menu. (If a command is highlighted in an open menu, pressing *Enter* chooses that command.)

Notice that each menu name has an underlined letter. As a shortcut to opening a menu, press *Alt* plus the underlined letter. For example,

Alt F — Opens the File menu.

Alt M — Opens the Form menu.

Alt T — Opens the Tools menu.

After you have opened the menu, highlight the menu command you want:

↑

or

↓

Highlights a command in an open menu. Or press the underlined letter in the command name to choose that command. For example, when the File menu is open, you can press *O* to open a file, *S* to save a file, or *X* to leave ObjectVision and unload it from memory.

Enter — Carries out a selected command or command button.

Spacebar — Activates a selected button on an application's form or checks a box when a value option is highlighted.

Esc — Exits from a menu without choosing a command.

Click a menu name to open it. Then click a command name to choose it.

As a shortcut, position the pointer on the menu name you want and drag to highlight the command. When you release the mouse button, the command is carried out.

Running applications

When the ObjectVision window isn't occupying the entire Windows desktop you can move the window by choosing Move. Press the arrow keys to reposition the window, then *Enter* when you finish.

Alt Esc — Move to another application on the Windows desktop, activating it if it is loaded in memory.

Alt Tab — Restore a minimized application to its previous size. Press *Alt+Tab* until the icon you want is highlighted, then release *Alt* only. After the application window opens, release *Tab*.

Ctrl Esc — Opens the Task List window so you can switch to another application loaded in memory.

ObjectVision Control menu commands

ObjectVision application window Control menu:

Alt Spacebar — Opens the Control menu. Use the arrow keys to select a command, then press *Enter* to choose it.

Alt F4 — Closes the application window. Windows will prompt you for confirmation if your current work is unsaved.

ObjectVision form window Control menu:

Alt - — Opens the Control menu. Use the arrow keys to select a command, then press *Enter* to choose it.

Ctrl F7 — Lets you move the form window with the arrow keys.

Ctrl F8 — Lets you resize the form window with the arrow keys.

Ctrl F4 — Closes the form window.

 — Double click the Control-menu box to close the application or form window. Windows prompts you for confirmation if your current work is unsaved.

Click any visible portion of another window on the desktop to switch to that other window. If no other windows are visible, resize the front-most window first.

Click the Maximize button to maximize the window.

Click the Minimize button to minimize the window.

If the window is open on the desktop, click the Restore button to return the window to its previous size.

If the icon is on the desktop, click the icon and choose Restore to return the window to its previous size. Or, double-click an icon to restore the application window to its previous size.

Getting Help online

Alt H — Opens the Help menu.

Help for a field (during form completion):

F1 — Displays Help for a highlighted field *if* the form designer created help for that field.

Esc *or* Alt F4 — Closes the Help window.

→ *or* ↓ — Scrolls down one line at a time.

← *or* ↑ — Scrolls up one line at a time.

PgUp *and* PgDn — Scroll the help window one screenful at a time.

Alt Spacebar — Opens the Help Control menu.

Alt F4 — Closes the Help window.

Ctrl Esc — Opens the Task List window so you can switch to another application loaded in memory.

Click outside the Help window to close it.

Drag the Help window border to resize it. (Move the pointer to the window's border until it changes to a double arrow.)

ObjectVision Help:

Alt H	Opens the Help menu.
F1	Opens the ObjectVision Help window (unless you're in form completion mode *and* the form designer created help for the selected field). If a command or a tool is highlighted, help for that topic appears; otherwise, the Help Index appears.
Alt Spacebar	Opens the Help Control menu.
Alt F4	Closes the Help window.
Ctrl Esc	Opens the Task List window so you can switch to another application loaded in memory.
Alt F4 *or* Alt F X	Closes the Help window.
Alt I	Displays the Help Index.
Alt B	Moves back to the previous help topic. Repeatedly pressing this shortcut moves back through all the topics viewed in the current session until Index was chosen. Choosing Index erases your Help history.
Alt R	Moves back to the previous help topic if the topic is a part of a browse sequence.
Alt O	Moves forward to the next help topic if the topic is part of a browse sequence.
Alt S	Displays the Search For dialog box which lets you locate help topics from a list of subjects.
↑ *or* ↓	Scrolls the help text one line at a time in the direction of the arrow key.
PgUp *or* PgDn	Scrolls the help text up or down one screenful at a time.

PgUp

and

PgDn

Scroll the help window one screenful at a time.

Tab

Selects the next underlined help topic.

Shift Tab

Selects the previous underlined help topic.

Enter

Chooses the selected help topic. If the topic has a solid underline, help for that topic appears. If the topic has a dashed underline, definition for that term appears onscreen as long as you hold down the key.

Drag the Help window border to resize it, or click the Minimize, Maximize, or Restore buttons.

Click a word that has a solid underline to jump to help for that topic.

Position the pointer on a word that has a dotted underline and hold down the mouse button to display a definition for that word.

Using dialog boxes

Esc

Closes the dialog box and leaves the settings unchanged.

Spacebar

Toggles a selected check box between checked and unchecked.

Enter

Activates a highlighted command button.

Tab

Moves to the next named option or group of options. Or, press *Alt* plus an underlined letter in the option name to select that option directly.

Shift Tab

Moves to the previous named option or group of options.

Ctrl Enter

Moves to a new line in a text box.

↓

or

↑

Selects a highlighted radio button.

↓ *or* ↑	Highlights a selection list item. Or, press the first letter of the item name to highlight the first item in the list beginning with that letter.

Completing forms

Alt -	Open the current form's Control menu.
Ctrl F4	Closes a form. Or, choose Close from the Control menu.
Ctrl F7	Lets you move a form with the arrow keys. Or, choose Move from the Control menu; then press the arrow keys to reposition the window and press *Enter* when you finish.
Ctrl F8	Lets you resize a Scratchpad form. Or, choose Size from the Control menu; then press the arrow keys to resize the form's border and press *Enter* when you finish.
Ctrl PgUp *or* Ctrl PgDn	Scrolls a form horizontally.
PgUp *or* PgDn	Scrolls a form vertically.

Fields

Moving between fields:

Alt Backspace	Backs up to the previous field and restores the previous value (only if pressed *immediately* after pressing *Enter* or *Tab* to enter a typed value).
Enter	Enters a typed value in the field and moves to the next field requiring user input.
Tab	Enters a typed value in the field and moves to the next field, going from left to right and from top to bottom.
Shift Tab	Moves to the previous field, going from bottom to top and from right to left.

Selecting field value options:

Spacebar — Activates a highlighted button or checks a highlighted check box.

Enter — Checks a highlighted check box. Highlight a check box by pressing ↓ or ↑ when the field is selected, or press the first letter of the response you want, repeating if necessary until you move to the response you want.

Enter *or* Tab — Selects a highlighted list item. Highlight the item you want by pressing the first letter of the item you want, or press ↑ and ↓.

Spacebar — Toggles a true/false field between checked an unchecked.

Editing text in fields:

Shift → *or* Shift ← — Highlights text. Highlighted text is replaced by the next characters you type.

Esc — Restores the previous field value (*only* if you are still in the field and haven't pressed *Enter* to enter the new value).

Home — Moves to the left of the first character in the field.

End — Moves to the right of the last character in the field.

Del — Erases the character immediately to the right of the pointer. Or, erases any highlighted text.

Backspace — Erases the character immediately to the left of the pointer. Or, erases any highlighted text.

Double-click any letter in a word to highlight the entire word.

Drag to highlight any amount of text in a field. Highlighted text is replaced by the next characters you type.

Viewing decision trees

Moving between nodes:

Home — Moves to the root node of the decision tree.

↑ — Moves to the previous node, if any, at the same nesting level in the decision tree.

↓ Moves to the next node, if any, at the same nesting level in the decision tree.

← Moves to the node, if any, at the preceding nesting level in the decision tree.

→ Moves to the first node, if any, at the subsequent nesting level in the decision tree.

Scrolling:

Ctrl PgUp Scrolls a tree horizontally.

or

Ctrl PgDn

PgUp Scrolls a tree vertically.

or

PgDn

Changing the display:

Ctrl Home Enlarges the size of the decision tree.

Ctrl End Reduces the size of the decision tree.

Form Tool

Control menu commands:

Alt - Opens the form Control menu.

Ctrl F7 Lets you move a form with the arrow keys. Or, choose Move from the Control menu; then press the arrow keys to reposition the form and press *Enter* when you finish.

↑ Moves a selected form object in the direction of the key.

↓

←

or

→

Shift ↑
Shift ←
Shift →
or
Shift ↓

Lets you position and size a new form object. First, position the pointer using the arrow keys, then press *Enter*. Press the arrow keys again to define the size and shape of the new object, then press *Enter*.

Ctrl F8

Lets you resize a form using the arrow keys. Or, choose Size from the Control menu; then press the arrow keys to resize the form's border and press *Enter* when you finish.

Shift
Arrow keys

Lets you resize a select form object. *Shift*+arrow keys move the object's lower right corner.

Scrolling:

Ctrl PgUp
or
Ctrl PgDn

Scrolls a form horizontally.

PgUp
or
PgDn

Scrolls a form vertically.

Selecting a form object:

Tab

Moves to the next object, going from left to right and from top to bottom.

Shift Tab

Moves to the next object, going from bottom to top and from right to left.

Ctrl Tab

Lets you select multiple objects on a form, going from left to right and from top to bottom.

Shift Ctrl Tab

Lets you select multiple objects on a form, going from bottom to top and from right to left.

Click to select a form object.

Shift
Click

Begins or ends a contiguous block selection of form objects. If the begin block selection is above the end block selection, the objects are selected going from left to right and from top to bottom. If the begin block selection is below the end block selection, the objects are selected going from right to left and from bottom to top.

Ctrl *Click* — Selects multiple, non-contiguous form objects.

Stack Tool

↑ *or* ← — Moves toward the top form in the stack.

↓ *or* → — Moves toward the bottom form in the stack.

PgUp *or* PgDn — Scrolls the stack display vertically.

A P P E N D I X

B

Application limits

This appendix lists limits and precisions for your ObjectVision applications. It also covers the About command on the Help menu, which displays memory use and other information about an open application.

Table B.1
ObjectVision limits and precisions

Element	Limit
Application memory use	262,144 bytes (256K)
Number of fields	Limited by memory
Number of forms	Limited by memory
Number of tree nodes	Limited by memory
Conclusions	4,096 characters
Conditions	4,096 characters
Field value	4,096 characters
Help text	4,096 characters
Text value	4,096 characters
Form name	254 characters
Field name	254 characters
Date range	1/1/1800 to 12/31/2099, inclusive
@Function arguments	14
Numeric values	18 digits, including the decimal point
Path name length	63 characters

Database file compatibility

This section lists the Paradox and dBASE application versions that ObjectVision is compatible with.

ObjectVision is 100% compatible with Paradox versions 3.0 and 3.5, and Paradox .DB files written by Quattro Pro. ObjectVision is also compatible with dBASE III, dBASE III+, dBASE IV, Clipper, Fox, Quattro Pro and other 100%-compatible .DBF files.

However, users should be aware that ObjectVision *doesn't* support dBASE IV's .MDX index file, Clippers .NTX index file, or Fox's .IDX file. Clipper does support the .NDX index file format as well as its own .NTX index file format, and ObjectVision supports the Clipper locking protocol on the database and .NDX index files so data can be shared with Clipper applications.

ObjectVision only supports Clipper's file locking for dBASE-compatible files. DBASE's network file locking is not supported because it opens files exclusively for the dBASE application opening the file.

Getting application information

Help | About lists the following information about an open application:

- number of forms
- number of fields
- number of tree nodes
- memory use

As you design your application, use Help | About to track its size. If it nears the maximum permitted memory use (262,144 bytes), you might consider dividing your application into two or more related applications. You can link your application segments with ObjectVision DDE links. DDE links are explained in detail in Chapter 10, "Linking to data files."

A P P E N D I X

C

The ANSI character set

The ANSI character set defines character codes for 256 characters. The ASCII character set is the same as the first 128 ANSI characters (0-127). The numeric keypad *must* be used when you create an ANSI character. To create one of the characters, hold down *Alt*, type 0 (zero), type the corresponding code number, and then release *Alt*.

Code	Char	Code	Char	Code	Char	Code	Char
0	*	32	(space)	64	@	96	`
1	*	33	!	65	A	97	a
2	*	34	"	66	B	98	b
3	*	35	#	67	C	99	c
4	*	36	$	68	D	100	d
5	*	37	%	69	E	101	e
6	*	38	&	70	F	102	f
7	*	39	'	71	G	103	g
8	**	40	(	72	H	104	h
9	**	41	)	73	I	105	i
10	*	42	*	74	J	106	j
11	*	43	+	75	K	107	k
12	*	44	,	76	L	108	l
13	**	45	-	77	M	109	m
14	*	46	.	78	N	100	n
15	*	47	/	79	O	111	o
16	*	48	0	80	P	112	p
17	*	49	1	81	Q	113	q
18	*	50	2	82	R	114	r
19	*	51	3	83	S	115	s
20	*	52	4	84	T	116	t
21	*	53	5	85	U	117	u
22	*	54	6	86	V	118	v
23	*	55	7	87	W	119	w
24	*	56	8	88	X	120	x
25	*	57	9	89	Y	121	y
26	*	58	:	90	Z	122	z
27	*	59	;	91	[	123	{
28	*	60	<	92	\	124	|
29	*	61	=	93	]	125	}
30	*	62	>	94	^	126	~
31	*	63	?	95	_	127	▮

Code	Char	Code	Char	Code	Char	Code	Char
28	▮	160		192	À	224	à
29	▮	161	¡	193	Á	225	á
30	▮	162	¢	194	Â	226	â
31	▮	163	£	195	Ã	227	ã
32	▮	164	¤	196	Ä	228	ä
33	▮	165	¥	197	Å	229	å
34	▮	166	¦	198	Æ	230	æ
35	▮	167	§	199	Ç	231	è
36	▮	168	¨	200	È	232	é
37	▮	169	©	201	É	233	ê
38	▮	170	ª	202	Ê	234	ê
39	▮	171	«	203	Ë	235	ë
40	▮	172	¬	204	Ì	236	ì
41	▮	173	-	205	Í	237	í
42	▮	174	®	206	Î	238	î
43	▮	175	¯	207	Ï	239	ï
44	▮	176	°	208	Ð	240	ð
45	‘	177	±	209	Ñ	241	ñ
46	’	178	²	210	Ò	242	ò
47	▮	179	³	211	Ó	243	ó
48	▮	180	´	212	Ô	244	ô
49	▮	181	µ	213	Õ	245	õ
50	▮	182	¶	214	Ö	246	ö
51	▮	183	·	215	▮	247	▮
52	▮	184	¸	216	Ø	248	ø
53	▮	185	¹	217	Ù	249	ù
54	▮	186	º	218	Ú	250	ú
55	▮	187	»	219	Û	251	û
56	▮	188	¼	220	Ü	252	ü
57	▮	189	½	221	Ý	253	ý
58	▮	190	¾	222	Þ	254	þ
59	▮	191	¿	223	ß	255	ÿ

A P P E N D I X

D

Configuring the Paradox Engine

ObjectVision uses the Paradox Engine to provide access to Paradox tables. This open architecture lets ObjectVision share data with Paradox and any other application using the Paradox Engine.

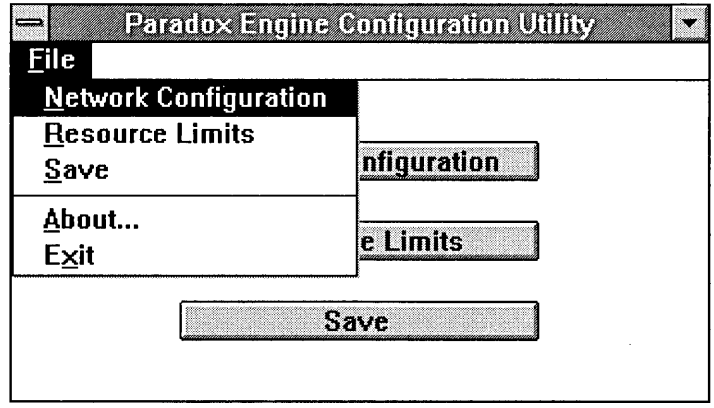
When you use ObjectVision to access Paradox data located on a network, or share local data with Paradox running in the DOS box, you *must* configure the local engine using the Paradox Engine Configuration Utility.

Using the Configuration Utility

The Paradox Engine Configuration Utility is a program called PXENGCFG.EXE that runs under Windows 3.0. It lets you modify settings in WIN.INI that control how the Paradox Engine operates.

Figure D.1 shows the Main menu window:

Figure D.1
CP Main menu



The File menu contains the following menu commands: Network Configuration, Resource Limits, Save, About, and Exit. Choose one of these commands by opening the File menu and choosing the command you want. Alternatively, you can also choose one of the buttons displayed on the Main menu: Network Configuration, Resource Limits, or Save. You can exit the Configuration Utility by choosing Exit from the File menu, or by double-clicking the Control-menu box in the upper left corner of the window.

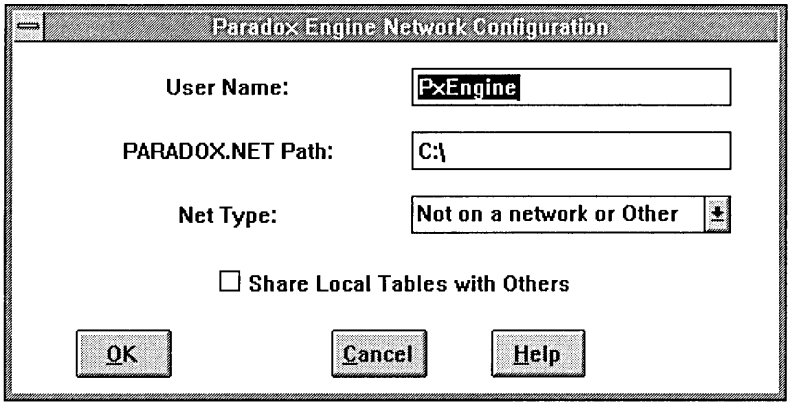
When you choose a command, you'll see a dialog box with the current values for specific variables. Edit these variables to suit your environment. Choosing Save makes your changes permanent. Closing the Configuration Utility without choosing Save leaves the original settings unchanged.

Network Configuration

The Network Configuration dialog box specifies the file-sharing characteristics of the tables accessed by the Engine.

Selecting on any line highlights the entire line for editing. Choosing the Help button offers an explanation of the currently highlighted line (its effects and legal values).

Figure D.2
The Network Configuration dialog box



- **User Name**: lets you enter a user name. If the Paradox Engine locks a table, other Windows applications attempting to access that table will be told, "Table *Name* is locked by user *UserName*." The current value is stored in WIN.INI; its default value is Vision.
- **PARADOX.NET Path**: lets you specify the location of the PARADOX.NET file. If users are using the Paradox Engine on tables located on a network server, they must obtain the location of PARADOX.NET from the network administrator. Users using the Paradox Engine on tables located on their own PC should use the default setting, C:\.
- **Net Type**: lets you specify your network type by selecting from a list:

  Not on a network or Other
  Novell NetWare
  3Com 3+
  3Com 3+Open
  IBM PC LAN
  StarGROUP (AT&T)
  Banyan

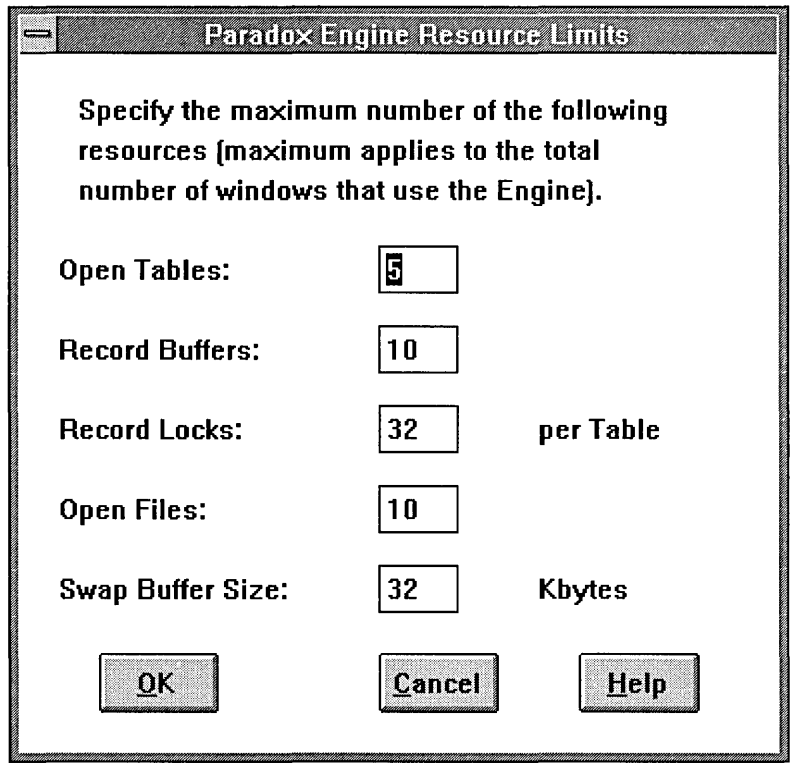
Warning! *Checking this box adversely affects performance, so only check it if you need to.*

- **Share Local Tables with Others**: check this box only if you're running either Paradox 3.5 or Engine applications concurrently from the DOS box in Windows. If you're simply running Engine applications in multiple windows, do not check this box.

Resource Limits

The Resource Limits dialog box sets the resources for an Engine Windows application.

Figure D.3
The Resource Limits dialog box



- **Open Tables**: sets how many tables can be open at once. The range is 1 to 64, the default is 5. As with all resource limits under Windows, this setting applies to the total number of open tables in all applications, in all active windows.
- **Record Buffers**: sets how many record transfer buffers are available at one time. The range is 1 to 128, the default is 10.
- **Record Locks**: sets the maximum number of record locks per table. The range is 1 to 128, the default is 32. Setting this option high decreases network performance.

- **Open Files**: sets the maximum number of file handles for all applications in all windows. The range is 2 to 255, the default is 10.
- **Swap Buffer Size**: determines the size (in kilobytes) of the swap buffer. The swap buffer functions as a cache. A larger swap buffer reduces disk access for tables, but uses up memory that Windows might otherwise allocate to other purposes. The range is 8K to 256K, with a default of 32K. The value entered is rounded to a multiple of 4K.

G L O S S A R Y

This chapter is an alphabetical list of key ObjectVision terms. All of these terms are described in greater detail in the sections of the manual in which they appear.

A

active form The form currently being completed; it appears in front of all other forms in the application.

active window The window you are currently working in; the one that always appears in front of any others and has a title bar highlighted with a different color or intensity.

argument Specific information required by an @function. Most @functions require at least one argument.

B

block selection In the Form Tool, lets you select multiple fields in order to perform editing operations, assign or revise properties, or reposition the selected fields as a group of objects.

branch A series of related calculation-logic segments or steps that lead to a decision tree's conclusion.

branch node A segment of calculation logic (simple or complex) that selects a node below it by evaluating its associated field value. The name of the tree's evaluated field appears inside the branch node.

button fields An ObjectVision field type that is evaluated as unactivated (false) or activated (true). Button fields can be automatically created by Object-Vision when a link is first created. Buttons fields can perform complex tasks. For example, button fields can be assigned a single conclusion node to activate an external link.

C

calculated field A field that uses underlying decision logic to produce its value. A calculated field is indicated by a solid outline with an inner dotted line when it is selected.

calculation logic An expression, operation, or constant in a field's decision tree that computes its value.

choose To execute or carry out a command, or to activate a command button.

circular logic When a conclusion expression refers to its associated field, it re-prompts the user for the field value. Circular logic is used to prompt the user when all other conclusions in the decision tree are inappropriate.

complex branch node An individual segment of calculation logic that requires evaluation of one or more decision trees from other fields. A complex branch node is indicated with a flow chart symbol to remind you of other, underlying logic.

concatenation The joining of two or more text strings into a single text string.

conclusion The segment or step of calculation logic that is evaluated to determine the resulting value of a field's decision tree. The conclusion expression is located in the end node for a logical path in a field's decision tree.

conclusion node The ending segment of calculation logic that provides a value to the field. A conclusion node is indicated by an inverted triangle placed to the left of the conclusion expression.

condition The segment or step of calculation logic that is evaluated to determine what node is selected next.

D

DateTimeNumber An argument that is a number in the range –36,522 (January 1, 1800) to 73,050 (December 31, 2099). The decimal portion of DateTimeNumber represents the time from 12:00 noon to 11:59:59 p.m. and is computed as a fraction of a 24-hour day.

DDE (Dynamic Data Exchange) The Windows protocol for dynamically transferring data between Windows applications. DDE links are read-only links that require the Windows application and its document file to work.

decision logic A process defined as a series of small steps, or nodes. Decision logic is graphically represented for each field's decision tree in the Tree Tool.

decision path The segments of calculation logic used to determine a field value. This path is indicated with a bold line when a field's tree is displayed using Field | Show Tree.

decision tree A graphical representation of ObjectVision's field-value calculation process that uses one or more segments of calculation logic, or nodes.

default A calculated field value that can be overridden by the user. In the *Myorder* application, you assigned the Item field a default value of Widget.

Dynamic Data Exchange (DDE) The Windows protocol for dynamically transferring data between Windows applications. DDE links are read-only links that require the Windows application and its document file to work.

E

Edit form The status of the active form in the Form Tool. The `(Edit)` status appears in the form's title bar to indicate that it receives the actions you perform.

empty node Appears when a field has no decision tree, or a branch node has no nodes under it. If an empty node is evaluated, an error value is returned.

expressions You use expressions to create complex mathematical, logical, string, or @formula operations for evaluating complex value combinations. For example, you might use expressions to multiply a series of numbers or get data from external data files.

external links The part of an ObjectVision application used to read from and write to ASCII, Paradox, dBASE-compatible, and Btrieve data files. Additionally, read-only links can be created for Dynamic Data Exchange (DDE) files. Links are created with the Tools | Links dialog box or using @functions.

F

field A uniquely named, blank object that contains values entered by a user or calculated or linked values. A field that is not put on any form automatically appears on the Scratchpad form.

field sequence The order of user movement between the fields on a form. The field sequence is determined by the physical ordering of the fields from left to right and top to bottom. The relative position of the field's bottom right corner determines whether it is before or after another field.

font A typestyle used for a field label or text object.

form A uniquely named object that contains a collection of fields, analogous to a paper form. A form is the primary interface between your ObjectVision applications and users.

Form Clipboard An ObjectVision clipboard for cutting, copying, and pasting forms in the Form Tool. Unlike the Windows Clipboard, the Form Clipboard can't be used to transfer data between different applications.

form completion When a user fills in an ObjectVision application's field values. After a form is completed, the status indicator in the title bar reads `[Complete]`.

Form Tool The window where the application designer can create and edit forms and their objects (text, fields, rounded and filled rectangles, lines, and graphics).

@function Performs advanced calculations and operations within a field's calculation logic. Built-in ObjectVision @functions are compatible with Quattro Pro's and typically require at least one argument, or function-specific information.

G

Goal form The top form in the application's stack of forms. The Goal form appears when a user first opens the ObjectVision application. When another form is selected by the user (interrupting guided completion) it becomes the Goal form.

graphic A Windows Paintbrush or Windows metafile image copied to the Clipboard. A graphic is pasted into ObjectVision from the Windows Clipboard using the Objects | Graphics command.

guided completion When a user presses *Enter*, only a field requiring user input is selected next. If a user interrupts this method of selection by selecting some other field, guided completion can be restarted with Edit | Resume.

H

handles In the Form Tool, small black squares in the corners of a selected object or at the ends of a selected line. You can select and reposition handles to change the object's shape or length.

L

label A constant string value that doesn't need to be enclosed in double-quotation marks.

label prefix A single-quotation mark (') used as the first character of an expression to force that expression to be evaluated as a label.

links The part of an ObjectVision application used to read from and write to to ASCII, Paradox, dBASE-compatible, and Btrieve data files. Additionally, read-only links can be created for Dynamic Data Exchange (DDE) files. Links are created with the Tools | Links dialog box or using @functions.

Links Tool The ObjectVision dialog boxes where you can create, modify, or delete links to external data files.

literal characters In a picture string, any number, letter or punctuation character that isn't a match (# ? & @ !) or reserved (* [] {} ,) character. To use a match or reserved character as a literal character in a picture string, precede it with a semicolon.

load statement The area in the Windows system file WIN.INI where you can instruct Windows to load ObjectVision whenever Windows is first loaded.

logical expressions A segment or step of calculation logic that evaluates as either 1 (true) or 0 (false). Logical expressions are typically used with @functions in conditional statements.

M

match characters In a picture string, the unique characters you use to define a kind of pattern for the value users type into a field. The match characters are

- # (digit only)
- ? (letter only)
- & (letter only, convert to uppercase)
- @ (any character)
- ! (any character, convert letters to uppercase)

maximize To enlarge the active window so it occupies the entire Windows desktop by clicking the Maximize button or choosing the Control | Maximize command.

minimize To reduce the active window to an icon on the desktop by clicking the Minimize button or choosing the Control | Minimize command. When an application is minimized, it is still loaded in memory.

multiple selection In the Form Tool, highlighting several objects so subsequent actions are carried out on them all at once.

N

nesting level In a decision tree, the number of positions away from the root node. Branches that are the same distance from the root node are at the same nesting level.

In an @function or expression, parentheses enclose operations to be performed independently, and the contents of the innermost set of parentheses are always evaluated first. For example, in the expression @INT(@MOD(@NOW,7)), the @NOW function is nested inside the @MOD function and is evaluated first to provide the single argument required by @MOD.

node An individual segment or step of calculation logic that is used to evaluate a field value. A node can be an empty node, a simple or complex branch node, a root node, or a conclusion node.

O

object Any element you can place on a form, such as text, a field, filled or rounded rectangles, a line or a graphic. Objects can be assigned different properties, or attributes. For example, there are fourteen options for fill pattern and four options for line weight.

operators Used to express a relationship (logical, mathematical, or string) between two or more values. The result of an expression depends on the order in which the operations are performed.

override Calculated fields that are unprotected let a user enter a new value. After a users overrides a calculated value, the field displays a dot pattern. The calculated value can be restored using Field | Calculate.

P

paste The action of transferring data from a Clipboard to the active window. Or, transferring list items to an expression using the Paste Function or Paste Field buttons.

picture A pattern you specify to control values that users type into a field during data entry.

picture string The pattern of literal, match, and special characters you type into the Field Type | Picture | Picture String dialog box to define a template or pattern for a field value entered by a user. For example, you could type a telephone number or Social Security Number pattern users would have to follow.

points A typographers measure of font size, roughly equal to $\frac{1}{72}$ of an inch. The default Label Font is 8-point Helvetica, and the value font is 10-point Courier.

precedence The order in which operations are evaluated in an expression. Certain operators are always evaluated after others, unless nested.

protection A field property you assign with Properties | Protection to keep users from changing the field value or viewing a decision tree.

R

reserved characters In a picture string, the unique characters you must precede with a semicolon when you want them to appear as literals in the field value. The reserved characters are * *[ ] { }* ,

root node The first (leftmost) segment of calculation logic in the decision tree. The root node can be either a branch node or a conclusion node.

run statement The area in the Windows system file WIN.INI where you can instruct Windows to load and then run ObjectVision whenever Windows is first loaded.

S

Scratchpad form The form ObjectVision automatically creates to display any field not on a form.

select To position the pointer on an item and highlight that item. The highlighted item will receive the next action that ObjectVision performs.

selected field A selected field can be indicated with a solid line, a solid line with an inner dotted line, or a solid line with a dot pattern over the field. Everything a user types appears in this area on the form.

In the Form Tool, a selected field object is surrounded with a dashed line and has small black squares on its corners (for a field) or on its ends (for a line).

simple branch node An individual segment of calculation logic that is not dependent on decision trees from other fields.

stack The form order in an ObjectVision application, as displayed in the Stack Tool. The Goal form is the top form in the stack.

Stack Clipboard A temporary holding place for forms you cut or copy in the Stack Tool. The Stack Clipboard is different from the Windows Clipboard, and can't be used to transfer data between two different ObjectVision applications.

stack order The arrangement of an application's forms, as displayed in the Stack Tool, with the Goal form at the top.

Stack Tool The window that displays the order of forms in your ObjectVision application and lets you add, copy, paste, delete, or rearrange forms.

status The application title bar displays the form name and the state, or condition of the form: `[Goal]`, `[Complete]`, `[Prompt]`, or `[Edit]`.

syntax The acceptable format for defining expressions or @functions in ObjectVision. For example, all expressions must begin with one of the following characters:

0 1 2 3 4 5 6 7 8 9 . + - (@

T

Title bar The highlighted horizontal bar at the top of a window. The title bar contains the name of the active application, form, or tool. Form title bars also contain the status of the form: `[Goal]`, `[Prompt]`, `[Complete]`, or `[Edit]`.

Tree Clipboard A temporary holding place for tree objects you cut or copy in the Stack Tool. The Tree Clipboard is different from the Windows Clipboard, and can't be used to transfer data to any other application.

Tree Tool The ObjectVision window where you can create or modify decision trees.

V

values The data contained in fields. A user can type values, select them from a list, or check them; an ObjectVision application can calculate values or read them from an external data file.

INDEX

C

E

N

O

P

T

REFERENCE

OBJECTVISION™
FOR WINDOWS

BORLAND

CORPORATE HEADQUARTERS: 1800 GREEN HILLS ROAD, P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001, (408) 438-5300.
OFFICES IN: AUSTRALIA, DENMARK, FRANCE, GERMANY, ITALY, JAPAN, NEW ZEALAND, SINGAPORE, SWEDEN AND THE UNITED KINGDOM
PART # 25MN-OBV01-10 ■ BOR 2101