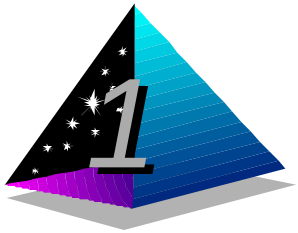
INTRODUCTION

Contents



This chapter provides:

- An introduction to Clarion for Windows.
- An overview of what you'll find in the *Getting Started* book.
- A “roadmap” guiding you to the information covered in the other books included with Clarion for Windows™.
- Typeface and other document conventions.
- A reminder about product registration.
- A summary of our technical support programs.
- Information about Clarion's fax-back system for quick technical support.

WELCOME TO THE FAST TRACK FOR WINDOWS APP DEVELOPMENT

Welcome to Clarion for Windows. You have just purchased what TopSpeed Corporation believes is *the most powerful Windows application development tool on the market!* You can now build sophisticated Windows applications faster than you ever thought possible. This revolutionary development environment will dramatically increase your productivity. The executable programs you create with it run as fast as those tediously crafted in languages such as C. You can connect to practically any existing database effortlessly.

Does this sound too good to be true? Just do the Quick Start Tutorial in chapter three of this book, and we’re sure you’ll be convinced in less than an hour.

Clarion for Windows is a very complete and comprehensive product; however, you can master it one step at a time. Before you know it, you’ll be developing Windows applications “at the speed of light.”

You now have both a flexible Rapid Application Development (RAD) platform, *and* the power of the underlying Clarion language to create complex Windows applications. Its *Point and Click* development process removes you from the complexity and tedium of typical Windows programming platforms.

Clarion for Windows' Application Generator builds fully customized Windows programs in a fraction of the time other programming environments require, yet *makes coding optional*. The template driven environment provides extensive code reusability without the steep learning curve of object oriented programming.

The underlying Clarion language is a powerful, yet easy to understand, business-oriented fourth-generation programming language (4GL). Combined with our high performance database drivers, Clarion for Windows delivers the shortest development cycle *and* fastest executable for *your* project.

- Template Driven Rapid Application Development

  Pre-written standard procedures—templates—provide customizable, reusable support for a wide range of functions such as browsers, forms, and reports. Just pick a template from a list, and fill in the prompts.

  The templates are fully customizable to the way you want your applications to be. You can easily add your own, or add other third party templates to your template suite!

- WYSIWYG (What You See Is What You Get) Formatters

  Use on-screen formatters to edit or add window and report controls to the template defaults. Easily customize list boxes, automatically associate database fields with entry controls, and choose actions for standard menu items, as you integrate them into the application.

  Essential procedures such as data validation and referential integrity checks—time-consuming tasks on other platforms—are coded for you *automatically*. Windows development projects which normally take months using other tools take a fraction of the time with Clarion.

- Immediate Results—Build *Your* Application in 1 Hour

  Clarion for Windows' QuickStart Wizard, QuickLoad, and all the Procedure Wizards will build a dictionary, choose templates, then create an application for updating, maintaining and reporting data within 60 minutes after you tear the shrink wrap off this package. You'll create the two *Quick Start Tutorial* applications in chapter three without writing a single line of code.

- 4GL Development—C Performance

  Unlike other runtime-interpreted RAD platforms, the executables you build with Clarion for Windows are *fast*. The TopSpeed compiler technology produces true Windows executables that *fly*.

  The Clarion language has the power and flexibility to support any application development project. It's a structured, compact, extensible language that supports the best of component-oriented development. Use Visual Basic Custom Controls (.VBX's) in your application. Create dynamic link libraries (DLLs) which applications written in other languages can call.

  For "hand-coders"—those writing apps with the Clarion language from scratch, rather than using templates—the development environment provides a wealth of tools. Window and Report formatters let you switch between graphically editing windows and reports, or editing their data structure declarations as text. The Text Editor provides color syntax highlighting. The Project Editor compiles and links at *TopSpeed*, and the world-class Clarion Debugger helps your project reach perfection.

- Database Independence

  New high performance database drivers support popular databases and accounting packages; including Xbase formats, Paradox, Btrieve, and others. On a local drive, they're far faster than the extra ODBC (Open DataBase Connectivity) layer many Windows database applications use (which Clarion for Windows supports as well).

  Store your data in (or convert existing data to) our proprietary *TopSpeed* file format for incredibly fast performance. It's efficient too; you can store multiple data files inside one physical DOS file, saving unnecessary use of end user file handles.

WHAT YOU'LL FIND IN THIS BOOK

The *Getting Started* book is your first introduction to the Clarion environment. The following lists the parts of this book and summarizes its content:

Introduction — *Chapter One*: the chapter you're reading now. Some of the material in this chapter also appears in the chapter of the same name in the User's Guide.

Setup

Chapter Two: the first step by step instructions in this book tell you how to install Clarion for Windows to your hard drive. Some of the material in this chapter also appears in the chapter of the same name in the User's Guide.

Turn to chapter two if you can't wait to try Clarion for Windows *now.*

Quick Start Tutorial

Chapter Three: a few quick steps with the Quick Start Wizard allow you create a complete application in *five* minutes.

Invest *less than an hour* and this chapter will show you how to use the Application Generator and the Wizards to create a simple application then extend its functionality into a sophisticated program that completely manages two related files.

You'll accomplish all this *without writing a single line of code*.

Development Flow

Chapter Four: an introduction to Clarion for Windows programming. It contains descriptions of the functional parts of the development environment, and an overview of the steps you take to create your applications. Some of the material in this chapter also appears in the chapter of the same name in the *User's Guide*.

Hands On Tutorial

Chapters five through sixteen: introduce the main development environment tools. It starts at the planning stages, then walks you through creating the data dictionary with the Dictionary Editor.

You'll load the Application Generator and work with Procedure, Control, and Code templates to produce an Order Entry application. You'll work with the Window Formatter to design windows. You'll work with the Report Formatter to design reports. You'll use the Text Editor to embed Clarion language source code into the code generated by the templates.

WHERE TO FIND MORE INFORMATION

There are four books and extensive on-line help available for Clarion for Windows. This provides a "roadmap," to show you where to look for information.

The *Getting Started* manual is the book you're reading now. It provides two step-by-step tutorials. The first utilizes the Clarion for Windows' Quick Start Wizard and Procedure Wizards to create an application *quickly*. The second "hands-on" tutorial provides a more extensive "walk through" the Development Environment, creating an Order Entry system.

The *User's Guide* provides a task-oriented description of the development environment, arranged by its major components. It describes the templates that ship with this product, and provides additional appendixes with information on topics such as the Clarion for Windows file drivers.

The *Language Reference* is the complete guide to the Clarion language. It provides descriptions of all statements and functions, with examples for each. The *Language Reference* is organized by functional topics.

The Clarion development environment also provides entry to the Windows Help system. The hypertext help appears when you press the F1 key, or choose one of the commands on the **Help** menu. The on line help is arranged by dialog box, to provide you with precise help text, when you need.

The full text of the *Language Reference* is on-line. When working with the Text Editor, place the insertion point on a Clarion language statement or function, then press the F1 key to view help for the item.

The *Template Language Reference* is the complete guide to Clarion's Template language. It provides descriptions of all statements and functions, with examples for each. The full text of the *Template Language Reference* is also on-line. This documents the Clarion Template Language, clearly demonstrating how to write your own templates. Access this document through the main table of contents for the help system.

Important: if any part of the on-line help text conflicts with the printed documentation, the on screen help takes precedence. TopSpeed Corporation makes every reasonable effort to ensure the printed documentation is up to date. However, lead-time required by printers may create a lag in the documentation; while we can update the help files that ship concurrently with a product revision, printed materials must 'catch up' later.

DOCUMENTATION CONVENTIONS

The documentation utilizes the typeface and keyboard conventions which appear below.

Typeface Conventions:

Italics	Indicates what to type at the keyboard, such as *Type This*.
SMALL CAPS	Indicates keystrokes to enter at the keyboard, such as ENTER or ESCAPE.
Boldface	Indicates commands or options from a pulldown menu or text in a dialog window. Note: this style also utilizes a different typeface to match the helvetica bold face which Windows uses as the system font.
LETTER GOTHIC	Used for diagrams, source code listings, to annotate examples, and for examples of the usage of source statements.

Keyboard Conventions:

F1	Indicates a keystroke. Press and release the F1 key.
ALT+X	Indicates a combination of keystrokes. Hold down the ALT key and press the X key. Then release both keys.

REGISTERING THIS PRODUCT

Before you begin using Clarion for Windows, fill out and mail in the registration card that came in the package. This Business Reply Card makes you eligible to receive several important benefits. Once registered, you can use TopSpeed's Technical Support services, and you automatically receive new product announcements and update alerts.

TECHNICAL SUPPORT

You can recieve unlimited free technical support for Clarion for Windows on CompuServe Information Service. Once connected to CompuServe, type GO TOPSPEED. TopSpeed employees, as well as TopSpeed Certified Support Partners (known as *Team TopSpeed*), will answer your questions in a timely manner. Additionally you will get advice and answers from other Clarion for Windows users. We strongly recommend that our customers take advantage of this service.

Paid telephone technical support is also available from TopSpeed Corporation. You can access our pay-per-call support by calling (900) 884-0444. Various paid support programs are also available. Call TopSpeed Corporation customer service at (800) 354-5444 or (305) 785-4555 for more information.

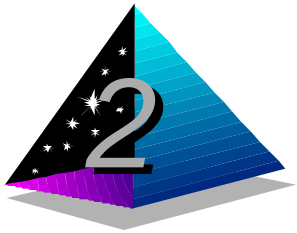
THE TOPSPEED FAX RETRIEVAL SYSTEM

TopSpeed Corp. also offers customers phone/FAX access to most often requested technical and marketing documents. Documents on-line include product brochures, technical documents, article reprints, price lists, and even a "What's Hot" update on TopSpeed products.

To request specific documents, dial (305) 785-4555, press 53, and listen for the FAX Retrieval System's instructions. The menu is interactive and user-friendly. First time callers can request a list of available documents to review before making a selection. You can then enter the document code number, and the material will be immediately delivered to you. You can access the system directly from your FAX machine or from any touch-tone phone.

Setup

Contents



This chapter explains the Clarion for Windows system requirements, setup process, and setup options.

SYSTEM REQUIREMENTS

You can run the Clarion development environment on any system that meets the minimum system requirements for Microsoft Windows 3.x, Windows 95™, or Windows NT 3.51.

- Windows 3.x, 8 Megabytes of RAM recommended.
- Windows 95, 12 Megabytes of RAM recommended.
- Windows NT 3.51, 16 Megabytes of RAM recommended.
- Minimum of 8 to 20 Megabytes free hard disk space, depending on the Setup options you select.

The applications that you develop with Clarion for Windows will execute comfortably on computers that meet only the minimum requirements for these operating systems.

THE SETUP PROGRAM

The Setup program, on disk one of your installation disks, decompresses and copies the Clarion for Windows files to your hard drive.

- For all the target operating systems, it provides you with options for installing the various components, such as the example files.
- In Windows 3.x, it *asks* before updating the PATH statement in your AUTOEXEC.BAT file to include the Clarion for Windows directory.

- In Windows 3.x, it installs Program Manager icons for the Clarion development environment, Debugger, Help files, and ReadMe files.
- In Windows 95, it installs the Clarion development environment, Debugger, Help files, and ReadMe file icons to the Start Menu programs.

Starting Setup

To start the Clarion for Windows Setup program in Windows 3.x:

1. Insert disk one of the installation disks into your floppy drive.
2. From Program Manager, File Manager, or other shell program capable of launching a program, choose **File ➤ Run**.
3. Type *A:\SETUP* (or *B:\SETUP*) in the **Run** dialog, and press the **OK** button.

   The Setup program provides an introductory screen and other text information. It prompts you for your dsesired Setup Options.

To start the Clarion for Windows Setup program in Windows 95:

1. Insert disk one of the installation disks into your floppy drive.
2. From the Start menu, choose **Settings ➤ Control Panel**.
3. Choose **Add/Remove Programs** then press the **Install** button.

   The Windows 95 Wizard will direct you through the installation process.

Setup Options

After starting Setup, you'll see a screen displaying a number of options.

1. Choose the Setup options by checking the boxes you want to install, then press the **OK** button.
2. Specify the target drive and directory, then press the **OK** button.

   Setup will install the main components of the Clarion Development Environment to a BIN subdirectory one level below the target directory you specify in the dialog.

   The Clarion for Windows Setup program installs *all* files to the target directory, and subdirectories beneath it. It installs *no* files to any other directory.

   During the installation, progress bars will display as Setup copies the files.

3. Choose **Yes** or **No** when Setup asks whether to modify the PATH for you.

   For Windows 3.x, the Clarion for Windows development environment requires that the BIN subdirectory be listed in the PATH environment variable. If you choose **No**, you *must* edit the AUTOEXEC.BAT file manually.

   The only other change to any of your system files is that Clarion for Windows adds its own section to WIN.INI (Windows' initialization file) when you run it for the first time.

4. Choose **Yes** or **No** when Setup asks whether to display the ReadMe file.

   If you don't wish to read it right away, you'll find an icon for it in the Program Manager group (or the Start menu) which Setup creates for you. We recommend reading it as soon as Setup has copied all the files.

5. Press the **OK** button when Setup is done.

STARTING CLARION FOR WINDOWS

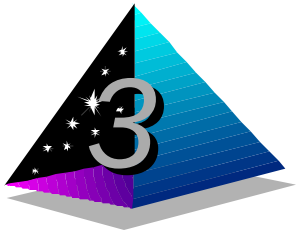
To run Clarion for Windows, locate the Clarion for Windows icon in the Clarion for Windows program group, and DOUBLE-CLICK on it:

The Clarion for Windows development environment appears, ready for you to begin work.

- You'll find a quick guide to the development environment parts—such as a diagram of the tool bar icons—in chapter four.
- Go on now to chapter three of this book to create *two* impressive applications—from data dictionary, to browse windows, to form windows, to *combination* browse and form windows—all in *less than an hour*.

QUICK START TUTORIAL

Contents



In Clarion for Windows, you can create a data dictionary and a working application—with no coding required. All you need to do is define the fields in your data file and the Quick Start Wizard creates a complete Windows application—in about five minutes if you're a fast typist! Your application has an update form procedure to maintain the file, a browse view on all the the sort keys and as many reports as there are keys.

Using the Quick Start Wizard, for each field in the data file you only have to define its name, display format picture, and key information—default values are provided for everything else. The Quick Start Wizard does everything else for you!

In this chapter:

- You'll use the Quick Start Wizard to create a Customer file and an application to maintain the file, then compile and run the program.
- You'll use Quick Load—it's another Clarion short cut that lets you quickly add a file to a data dictionary.
- You'll define the relationship between the two files in the Dictionary Editor, complete with Referential Integrity rules.
- You'll use the Application Generator to add more functionality to the application using one of Clarion's procedure Wizards.
- Finally, you'll use the Application Wizard to create a second complete relational application from the same data dictionary.

This should all take about thirty to forty five minutes—without *any* "coding" on your part. By the end of this chapter, you'll have two complete applications for a database containing two related files.

If you can do this much with Clarion for Windows' "short cuts," what can you expect to accomplish with all the other tools it provides? The remaining chapters in this book will teach you how to create an application "from scratch" using the Dictionary Editor, Application Generator, and all the other application design tools.

Welcome!

USING THE QUICK START WIZARD

To start, you should create a subdirectory for your application named C:\CW15\TUTORIAL. This assumes that you installed Clarion for Windows in C:\CW15. If you used a different directory, you will have to modify the instructions accordingly.

❑ Begin the Application.

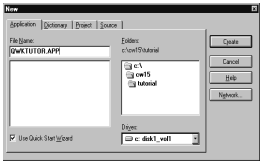
1. From Windows 3.1 **File Manager,** highlight the **CW15** subdirectory and choose **File ➤ Create Directory**. In the **Name** field, type *TUTORIAL*, then press the **OK** button. Return to Clarion for Windows.

   OR

   From Windows95 **Explorer,** choose **File ➤ New ➤ Folder**. In the entry field that appears, type *TUTORIAL*, then press ENTER. Return to Clarion for Windows.

2. Choose **File ➤ New.**

   The **New** dialog appears. It's a standard windows *Open File* dialog, allowing you to change the directory and type in the filename.

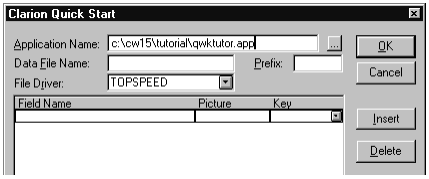


3. Select the *Application* tab.
4. Select the \CW15\TUTORIAL directory from the Directories list.
5. Type *QWKTUTOR* in the **File Name** field.

   The Quick Start Wizard uses this name as both the application name and dictionary name. This creates *QWKTUTOR.DCT* (the data dictionary file) and *QWKTUTOR.APP* (the application file).

6. Check the **Use Quick Start Wizard** box, then press the **Create** button.

   The **Clarion Quick Start Wizard** dialog appears.



❑ Define the data file.

1. Type *Customer* in the **Data File Name** field, and press TAB.

   Quick Start Wizard uses this name as the data file name. Notice that it takes the first three letters, *CUS*, and places them in the **Prefix** field. A file's Prefix makes field names unique across multiple files which share common field names.

2. Press TAB to accept *CUS* as the **Prefix.**

   Next, you choose a **File Driver**. This field defaults to **TOPSPEED** (one of Clarion's two proprietary file systems).

4. Press TAB to accept **TOPSPEED** as the **File Driver.**

   This takes you to the list box where you define the fields.

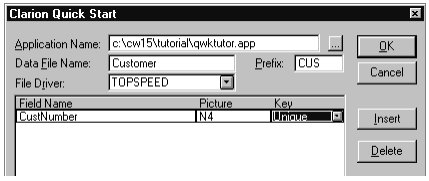
   Attention: Under Windows 3.1 you must have SHARE.EXE or the Windows VSHARE.386 driver loaded to use the TopSpeed file driver.

5. Type *CustNumber* in the first row of the **Field** column, and press TAB.

   This creates a field named *CustNumber.* Quick Start Wizard also uses this name for the default Prompt and Column Heading. The Prompt is used whenever you place the field on a window. Column Headings are used in reports.

6. Type *N4* in the **Picture** column, and press TAB.

   This specifies default display picture for window and report controls, which implicitly declares the data type for the Quick Start Wizard. In this case the data type is a four-digit DECIMAL field.



7. In the **Key** column, press the DOWN-ARROW to display the choices. Highlight *Unique,* then press TAB.

   This specifies that a unique key will be created with this field as its component. By specifying *Unique,* your application will ensure that each customer has a distinct number. Quick Start Wizard creates browses and reports based on every key created. This key will scroll through records sorted by *CustNumber*.

   The cursor is automatically positioned on the next row, allowing you to define the next field.

8. Type *Company* in the **Field** column, and press TAB.

   This creates a field named *Company* to use as the Customer's company name.

9. Type *S20* in the **Picture** column, and press TAB.

   This specifies the default display pictures for window and report controls. In this case, the data type is a twenty-character STRING field.

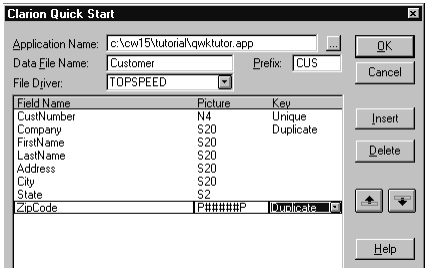
10. In the **Key** column, press the DOWN-ARROW to display the choices. Highlight *Duplicate,* then press TAB.

    This specifies a key will be created that allows duplicate entries. This allows you to have two Customers with the same company name.

11. Finish defining the file by creating the remaining fields, following this table:

FIELD	PICTURE	KEY
FirstName	S20	(no key)
LastName	S20	(no key)
Address	S20	(no key)
City	S20	(no key)
State	S2	(no key)
ZipCode	P#####P	Duplicate

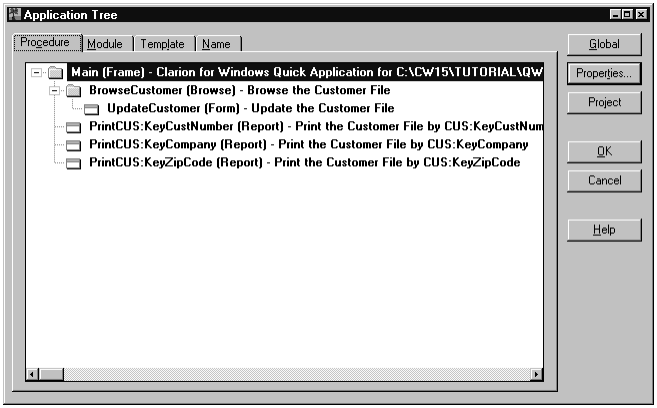
Notice the picture (P#####P) specified for the ZipCode field. It is a pattern picture that formats the digits for a five-digit U.S. zip code.



❑ Finish Quick Start Wizard and create the application.

1. When you have defined all the fields, press the **OK** button.

   Quick Start Wizard first asks if you are done defining fields. When you press the **OK** button, it then creates your application and displays the **Application Tree** dialog**.**



Notice the structure of the application. At the top of the tree is the *Main* (Frame) procedure, which creates an MDI (Multiple Document Interface) application frame. This procedure contains the menu which calls the other procedures.

Below the *Main* Procedure there is a *BrowseCustomer* (Browse) procedure with an *UpdateCustomer* (Form) procedure attached. The browse displays the data based on all three keys you specified when defining the fields. You also see three reports based on those keys.

The application is complete, without requiring any more information from you.

The Run Button

The Clarion for Windows toolbar icons in order: Pick, New, Open, Save. Following the gap: Make, Run, Debug.

2. Choose **Project ➤ Run** (or press the *Run* button on the toolbar).

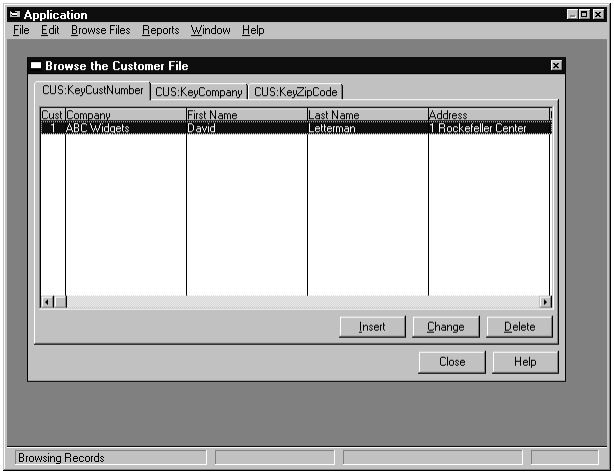
   The Application Generator creates the source code for your application, then it compiles, links, and executes.

   Congratulations! Your application is running. Now you can explore the application.

   Add some records. To do so, choose **Browse Files ➤ Browse the Customer file**, then press the **Insert** button. Type in the customer data and close the form window. Look at the different sort orders in the browse procedure, then resize the display columns and use the horizontal scroll bar to examine data in the fields that do not appear in the list initially. You can even print the reports if you want.

   When you are finished, exit the application. You can also try running the program from the File Manager or Explorer; it's a true Windows executable that does not require the Clarion environment to run.

3. Choose **File ➤ Exit** to exit your application.



Now we will save your work and exit from the Application Generator to modify the data dictionary we just created.

4. Choose **File ➤ Close** to close the Application Generator.

Modifying your Dictionary

In this section, we will make a simple dictionary modification to add some additional functionality to the application. We will enable the customer number to automatically increment each time you add a record.

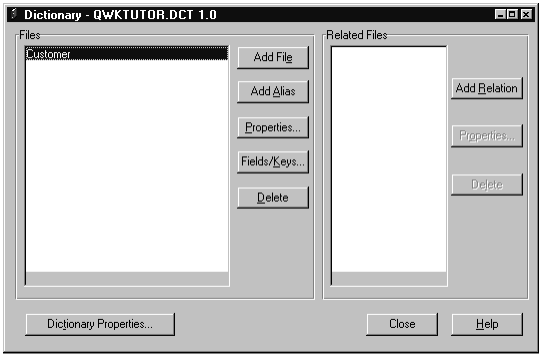
❑ Use the Dictionary Editor.

1. Choose **File ➤ Open** (or press the *File Open* button on the toolbar)**.**

   The **Open** dialog appears.

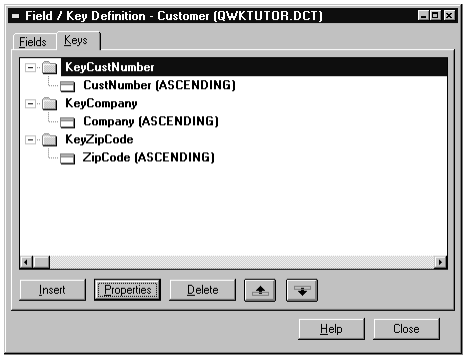
2. Choose the **Dictionary** tab, select the *QWKTUTOR.DCT* file, then press the **Open** button.

   This opens the data dictionary in the Dictionary Editor.



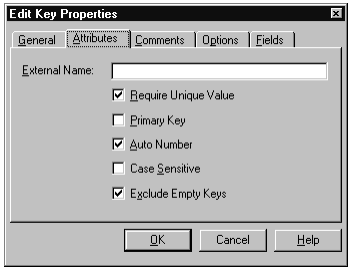
3. With the *Customer* file highlighted, press the **Fields/Keys** button.

   The **Field / Key Definition** dialog appears, displaying all the fields and keys in the file in two tabs.



4. Select the **Keys** tab, CLICK on *KeyCustNumber* to highlight it, then press the **Properties** button.

   The **Edit Key Properties** dialog appears.



5. Select the **Attributes** tab, check the **Auto Number** box, then press the **OK** button.

   Auto Number tells the Application Generator to add code to your application to automatically increment the value of the last key field component, ensuring it is a unique value for each new record.

6. In the **Field / Key Definition** dialog, press the **Close** Button.

7. Press the **Close** Button in the **Dictionary** dialog. When prompted, press the **Yes** button to save the dictionary file.

   That's all it takes. Next, we will re-generate the application to implement our modification.

Updating the Application

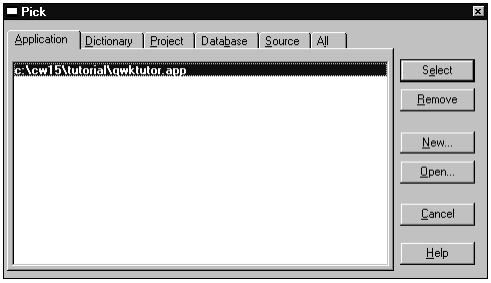
The change we just made in the dictionary is automatically incorporated into the program when you regenerate the source code.

We will reopen the application; this time using the Pick List.

❑ Use the Application Generator.

1. Choose **File ➤ Pick** (or press the *Pick* button on the toolbar).

   The **Pick** dialog displays a list of the most recently used files. This allows you to quickly reopen files.



2. Select the **Application** tab, highlight *c:\cw15\tutorial\qwktutor.app*, then press the **Select** button.

   Your application is reopened.

3. Choose **Project ➤ Run** (or press the *Run* button on the toolbar).

   The Application Generator generates the modified code for your application then compiles, links, and executes.

   With your application running, take a few minutes now to enter some test data. Notice that each time you add a new customer, the CustNumber field value automatically increments. When you are finished, you can close the program and exit Application Generator.

4. Choose **File ➤ Exit** to exit your application.

5. Choose **File ➤ Close** to close the Application Generator.

   Next, we will add a second, related file to the data dictionary.

ADDING A FILE WITH QUICK LOAD

You can add a file to the data dictionary quickly using the Quick Load option in the Dictionary Editor. Quick Load functions in a similar manner to the Quick Start Wizard—define the field (names, pictures, and keys), and a file definition is created in the dictionary. This option is available every time you add a new file to a dictionary.

In this section, we will add a file to store the phone numbers for customers. This will allow us to have many phone numbers for each customer (creating one-to-many relationship).

❑ Use Quick Load in conjunction with the Dictionary Editor.

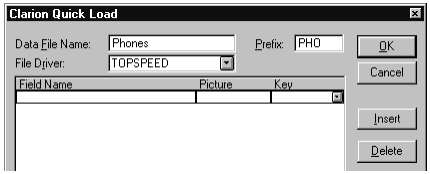
1. Choose **File ➤ Pick** (or press the *Pick* button on the toolbar).
2. Select the **Dictionary** tab, highlight *c:\cw15\tutorial\qwktutor.dct*, then press the **Select** button.

   The **Dictionary Editor** dialog appears.
3. Press the **Add File** button.

   A dialog appears asking, "Do you want to use Quick Load?"

4. Press the **Yes** button.

   The **Quick Load** dialog appears. Notice that it is very similar to the Quick Start Wizard dialog.



5. Select the **Data File Name** field, type *Phones* in it, then press TAB.
6. Press TAB to accept *PHO* as the **Prefix**.

   Next, you are prompted to choose a File Driver.
7. Press TAB to accept **TOPSPEED** as the **File Driver**.

   This takes you to the list box where you define the fields.
8. Type *CustNumber* in the first row of the **Field** column, and press TAB.

   This creates a field named *CustNumber.* This field is the link to the *CustNumber* field in the *Customer* file. Using the same name makes it easier to link the two fields in a relationship. In the generated code adding the prefix or the filename (separated by a colon) to field labels creates unique names for fields with the same name in separate files.
9. Type *N4* in the **Picture** column, and press TAB.
10. In the **Key** column, press the DOWN-ARROW to display the choices. Highlight *Duplicate*, then press TAB.

This specifies a key that allows duplicate entries, enabling you to have more than one record for each customer. This allows us to create a One to Many relationship between the two files.

The cursor is placed on the next row, allowing you to define the next field.

11. In the **Field** column on the next row, type *Area*, then press TAB.

12. In the **Picture** column, type P(###)P, then press TAB twice (don't define a key) to move on to define the next field.

    This formats the field as a three digit number, surrounded by parentheses (standard notation for telephone area codes in North America).

13. Complete the file by creating the remaining fields, following this table:

FIELD	PICTURE	KEY
Phone	P###-####P	(no key)
Description	S20	(no key)

14. When you have defined all the fields, press the **OK** button.

    Quick Load first asks if you are done defining fields. When you press the **OK** button, it then creates your file definition and adds it to the dictionary.

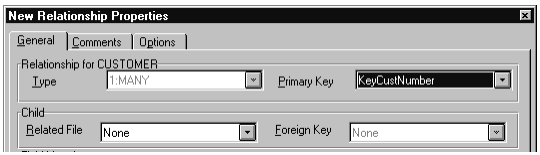
Adding a Relationship

Obviously, we want the Phones file to contain the phone numbers of the Customers. This means there must be a relationship between the two files. In this case, one Customer can have many Phones, making this a "One to Many" relationship. To define this relationship, we must link the files together in the data dictionary to provide the Application Generator with the information necessary to access the related records.

❑ Set the Relationship for the two files.

1. Highlight *Customer* in the **Files** listbox, then press the **Add Relation** button.

   The **New Relationship Properties** dialog appears. This is where you define the relationship.



2. Make sure the **Type** field is set to 1:MANY.

3. In the **Primary Key** field, press the DOWN-ARROW key to display the choices, highlight *KeyCustNumber*, then press TAB.

   This is the key on the Customer file (the One side of the relationship) that will be used to link the two files.

4. In the **Related File** field, press the DOWN-ARROW to display the choices, highlight *Phones*, then press TAB.

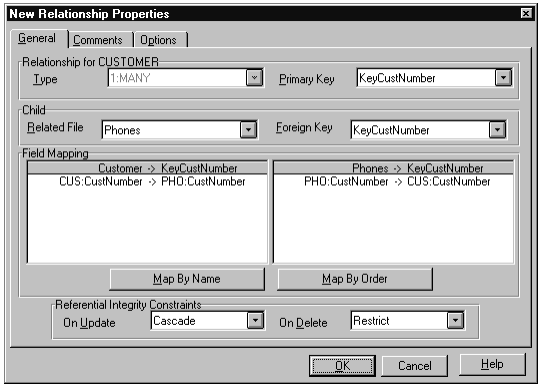
5. In the **Foreign Key** field, press the DOWN-ARROW to display the choices, highlight *KeyCustNumber*, then press TAB.

   This is the key on the Phones file (the Many side of the relationship) that will be used to link the two files.

   Next, the linking fields in the keys must be "mapped" so the Application Generator can know exactly which fields in the two files are related to each other. Since we used identical field names, this is easy.

6. Press the **Map By Name** button.

   The linking fields between the two files appear in the two **Field Mapping** list boxes.



❑ Set Up the Referential Integrity Constraints

1. Choose **Cascade** from the **On Update** dropdown list in the **Referential Integrity Constraints** group box.

2. Choose **Restrict** from the **On Delete** dropdown list in the **Referential Integrity Constraints** group box.

   The generated source code will automatically maintain referential integrity between the files. Chapter five explains this further. See *Using the Dictionary Editor* in the *User's Guide* for a brief discussion of relational database theory.

3. Press **OK** to close the **New Relationship Properties** dialog.

4. Choose **File ➤ Save**, or press the *Save* button on the toolbar to save the data dictionary.
5. Choose **File ➤ Close**, or press the **Close** button in the **Dictionary** dialog to close the dictionary.

PROCEDURE WIZARDS

In addition to the Quick Start Wizard, Clarion for Windows also has procedure Wizards that create Browse, Form, or Report procedures for you just as easily as the Quick Start Wizard creates an entire data dictionary and application. The Browse, Form, and Report Wizards are available to you whenever you create a new procedure in an existing application.

Now we will use the Browse Wizard to create a browse procedure for the Phones file. It will also create the update procedure at the same time.

❏ Load the Application Generator.

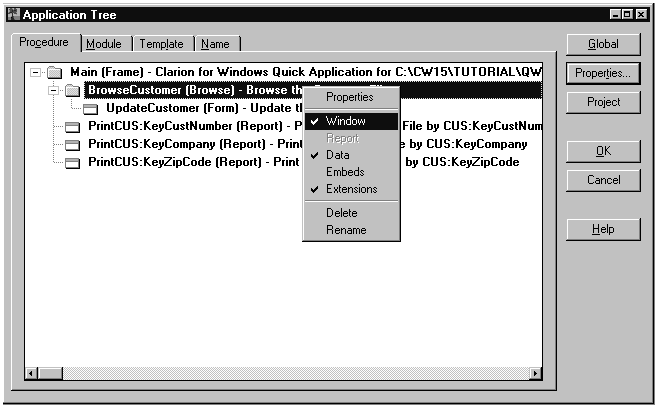
1. Choose **File ➤ Pick** (or press the *Pick* button on the toolbar).
2. Select the **Application** tab, highlight *c:\cw15\tutorial\qwktutor.app*, then press the **Select** button.

   The **Application Tree** dialog appears.

❏ Edit the Browse Procedure

1. Highlight BrowseCustomer (Browse) in the Application Tree then RIGHT-CLICK to display the popup menu.

   This menu allows you direct access to the Application Generator's tools without requiring you to open the procedure's Properties dialog first.



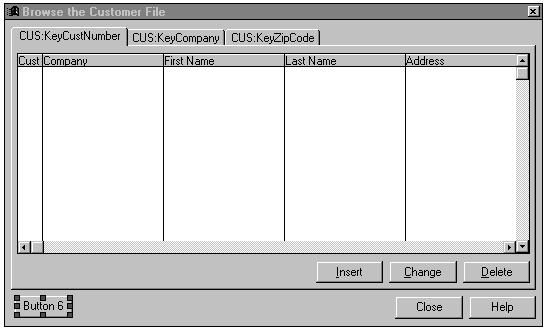
The popup menu displays a check mark beside the tools that contain data.

2. CLICK on **Window**.

   The Window Formatter appears. Here you can visually edit the window and its controls.

3. Choose **Control ➤ Push Button**, or CLICK on the Button tool in the Controls toolbox (the one with the icon that looks like an **OK** button).

4. CLICK near the bottom left corner of the window to place the new Button control below the tab sheets.



5. With the new button selected, RIGHT-CLICK to display the popup menu then choose **Properties** to call the **Button Properties** dialog.

6. Type *Phones* in the **Parameter** field.

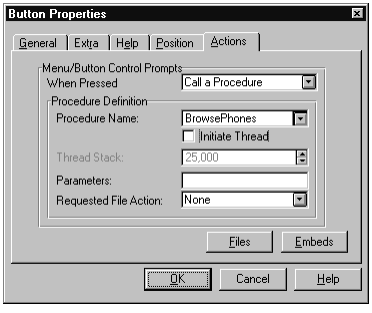
   This changes the text that will appear on the button face.

7. Select the **Actions** tab.

   This tab is where you specify what the control does. In this case, we want it to call a Browse procedure to display the Phones file records related to the currently highlighted Customer file record.

8. Choose **Call a Procedure** from the **When Pressed** dropdown list

   The Procedure Definition group box appears on the Actions tab.



9. Type *BrowsePhones* in the **Procedure Name** field then press the **OK** button.

   This names the procedure called when the user presses the button.

10. Choose **Exit!** from the menu to return to the Application Tree and press the **Yes** button when asked to save the window changes.

    The BrowsePhones procedure appears in the tree as a (To Do) item.

Using the Browse Wizard

Procedure Wizards generate entire procedures based on minimal information that you provide in response to a series of Wizard dialogs that ask for one piece of information at a time.

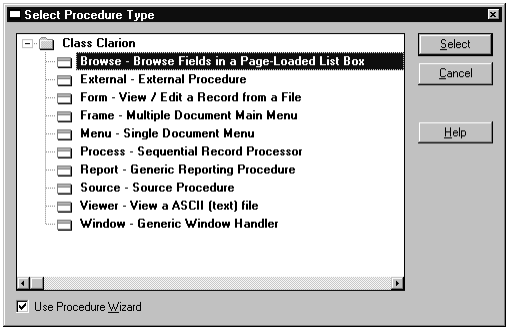
❑ Place the Control Template

1. DOUBLE-CLICK on the *BrowsePhones (ToDo)* procedure.

   The **Select Procedure Type** window appears. This allows you to choose the procedure template that the Application Generator uses to create source code.

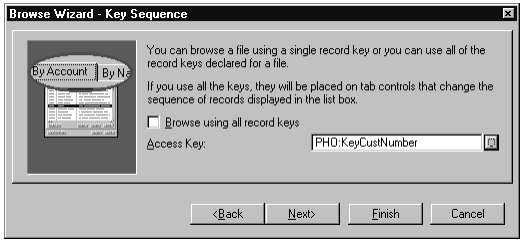
2. Highlight the **Browse** procedure Template, check the **Use Procedure Wizard** box, then press the **Select** button.

   Checking the **Use Procedure Wizard** box is the key to using the procedure Wizards.



3. After you read the first Wizard dialog's information, press the **Next** button to begin creating the procedure.

4. Type *Phones* in the entry control (or press the **...** button and select it from the list) in response to the question "Which file do you want to browse?" then press the **Next** button.

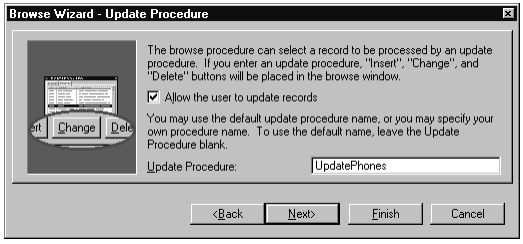


5. Uncheck the **Browse using all record keys** box, press the **...** button and select *PHO:KeyCustNumber* from the list, then press the **Next** button.

   This specifies the key used to sort the records for display in the list box.

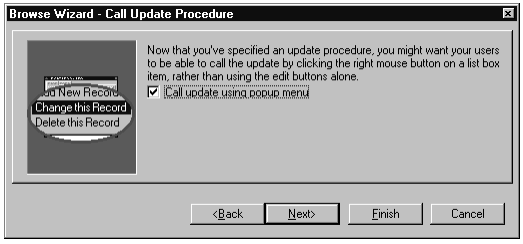
6. Type *UpdatePhones* in the **Update Procedure** entry box then press the **Next** button.

   This names the procedure that the Browse Wizard will create to update the Phones file records.

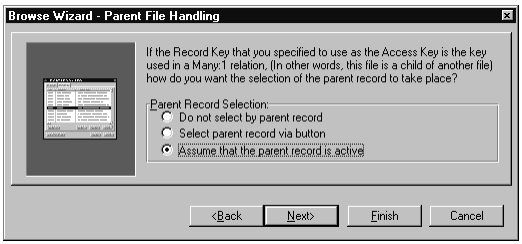


7. Check the **Call update using popup menu** box then press the **Next** button.

   This gives users the option of using a popup menu (that appears when they RIGHT-CLICK on the list box) to call the update procedure.

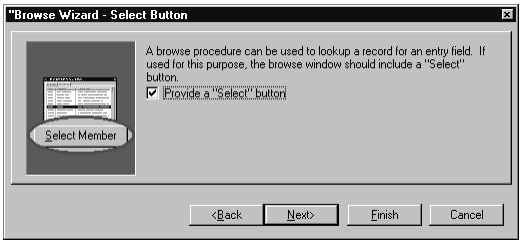


8. Uncheck the **Provide buttons for child files** check box then press the **Next** button.



9. Select the **Assume that the parent record is active** radio button then press the **Next** button.

   This ensures that the procedure is set up to only display the *Phones* file records that are related to the *Customer* file record currently in memory at the time the procedure is called.

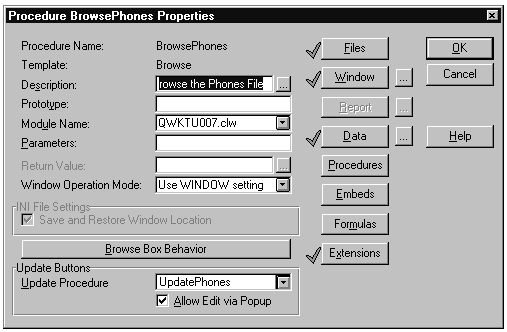


10. Uncheck the **Provide a "Select" button** box then press the **Next** button.

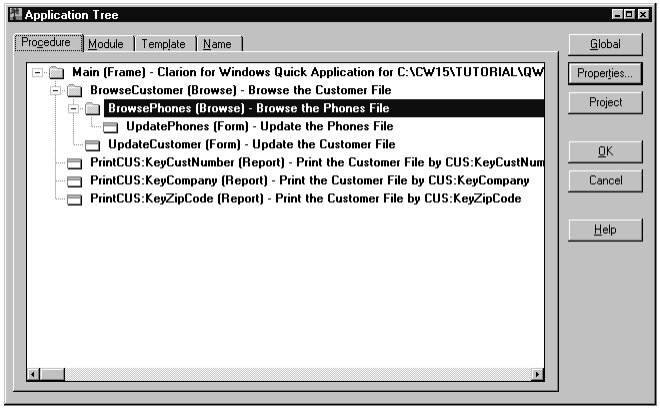
    Since this procedure will not be used to lookup data for an entry field, the Select button is unnecesary.

11. Press the **Finish** button to accept the default on the **Overwrite existing procedures** check box.

    The Browse Wizard now creates the Browse procedure for the Phones file and its associated update (Form) procedure to maintain the Phones file records. When finished, it leaves you on the new Browse procedure's **Procedure Properties** window.



12. Press the **Ok** button to return to the Application Tree.



Notice the two new procedures on the Application Tree that were created for you.

13. Choose **Project ➤ Run** (or press the *Run* button on the toolbar).

    Now you have a complete relational database application, featuring multiple browse orders on the parent file, browsing on the child file limited to the range of records related to one parent record, and Referential Integrity code to ensure that your database cannot ever be corrupted with "orphan" child records.

    Push the new Phones button and add some records to the Phones file.

14. When you have finished examining your new application, choose **File ➤ Exit** to return to Clarion for Windows.

15. Choose **File ➤ Close** to close the Application Generator.

APPLICATION WIZARD

Quick Start Wizard and the Browse, Form, and Report Wizards are all powerful tools. However, Clarion for Windows has another, even more powerful, Wizard in its arsenal of super-productivity tools.

The Application Wizard builds a complete application from an existing data dictionary. The data dictionary it uses may contain as many files as you like. The files should be fully defined, complete with all the file relationships, Referential Integrity, and Data Integrity rules that the Dictionary Editor allows you to defne.

To facilitate using the Application Wizard, the Dictionary Editor has property options on each file, field, and key that are expressly designed to interact with the Application Wizard (and all the procedure Wizards) to allow you to customize the application it creates for you.

❑ Use the Dictionary Editor to set Wizard options.

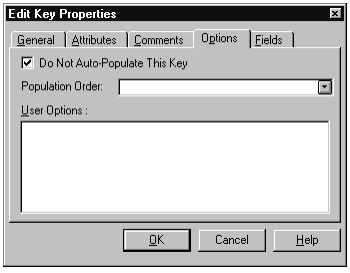
1. Choose **File ➤ Pick** (or press the *Pick* button on the toolbar).
2. Select the **Dictionary** tab, highlight *c:\cw15\tutorial\qwktutor.dct*, then press the **Select** button.

   The **Dictionary Editor** dialog appears.
3. Highlight the *Customers* file then press the **Fields / Keys** button.
4. Select the **Keys** tab, CLICK on *KeyCustNumber* to highlight it, then press the **Properties** button.

   The **Edit Key Properties** dialog appears.
5. Select the **Options** tab.

   The settings on this tab are all used to specify the behavior of the Wizards. The options in the upper half of the window are used with the standard set of Templates that Clarion ships. The **User Options** entry box is for the use of third-party add-on templates, so they may allow you to specify any constraints or options for their Wizards that are not covered by the standard options.
6. Check the **Do Not Auto-Populate This Key** box.

   Checking this box tells the Application Wizard not to use this sort key when creating a Browse procedure for the file. This will eliminate the *CUS:KeyCustNumber* tab on the Customer file's Browse procedure.



7. Press the **OK** button.
8. In the **Field / Key Definition** dialog, press the **Close** Button.
9. Press the **Close** Button in the **Dictionary** dialog. When prompted, press the **Yes** button to save the dictionary file.

   That's all it takes. Next, we will let the Application Wizard generate the entire application.

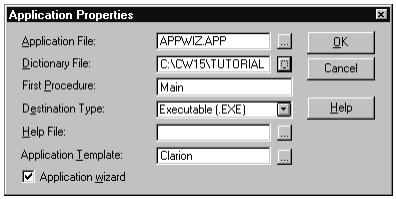
Using the Application Wizard

1. Choose **File ➤ New.**

The **New** dialog appears.

2. Select the *Application* tab.
3. Select the \CW\TUTORIAL directory from the Directories list.
4. Type *APPWIZ.APP* in the **File Name** field.
5. Uncheck the **Use Quick Start Wizard** box, then press the **Create** button.

   The **Application Properties** dialog appears.

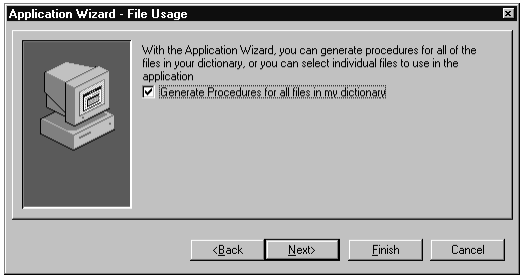


6. Press the **...** button to the right of the **Dictionary File** entry box.

   The **Select Dictionary** dialog appears.

7. Highlight the *c:\cw15\tutorial\qwktutor.dct* file and press the **OK** button.
8. Check the **Application Wizard** box, then press the **OK** button.

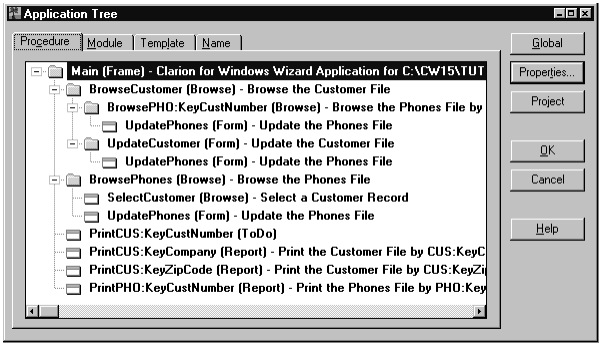
9. Press the **Next** button.



10. Press the **Next** button.

11. Press the **Finish** button.

The Application Wizard creates the application.



12. Choose **Project ➤ Run** (or press the *Run* button on the toolbar).

Congratulations! Your second complete application is running. Now you can explore it and compare it to the application you already created.

What's Coming Next

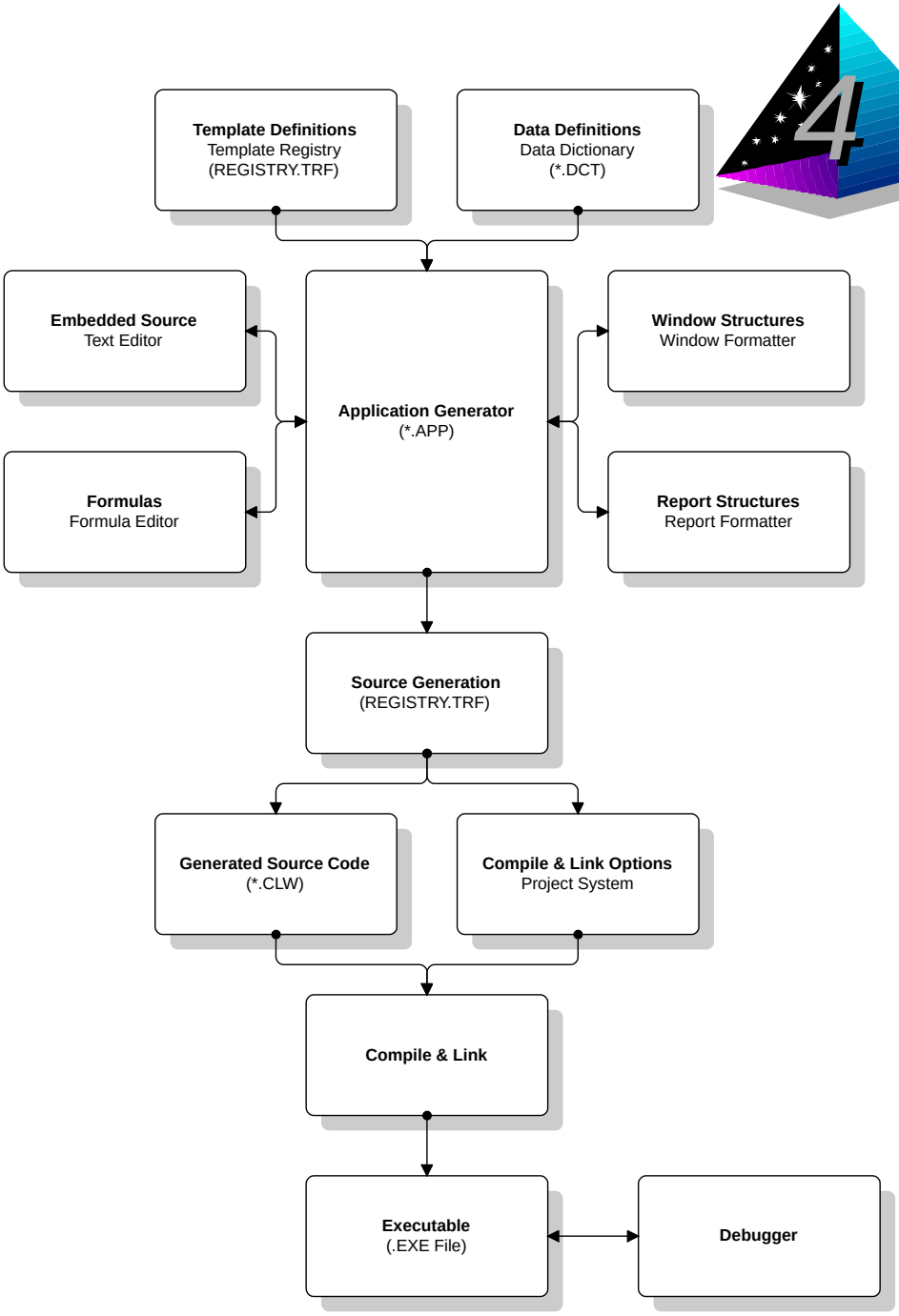
Besides all the Wizards, Clarion for Windows puts at your disposal a host of other productivity tools that you can use in the Application Generator, starting with the Control, Extension, and Code Templates that can add functionality to any procedure. These tools make modifying and customizing procedures just as easy as using the Wizards—and *none* of them require that you write any code!

The remainder of the *Getting Started* book walks you through creating an Order Entry system "from scratch" without using the Wizards. This will not only demonstrate all the other tools, but will also show you how much power you have when you finally do write a little source code yourself to provide some "non-standard" functionality.

Continue on. You've only just skimmed the surface of Clarion for Windows, *and there's a lot more!*

DEVELOPMENT FLOW

Contents



Clarion for Windows is an entirely new way to create Windows applications—as the application that you just created in less than an hour of your time demonstrated! However, Clarion is much more than the Wizards.

The foundation of Clarion is a fourth-generation, business oriented programming language. As implemented in Clarion for Windows, the language automatically handles the extra "housekeeping," such as checking the application's message queue. Many other Windows programming languages normally require many lines of code and complicated case statements for handling such chores. File driver independence is built into the language; Clarion for Windows contains dynamic link library drivers for most popular PC database formats.

On top of the Clarion language is the development environment—a set of tools that are specially designed for Rapid Application Development (RAD). The Application Development Flow chart on the previous page graphically depicts how the working parts of the development environment connect with each other when you use the Application Generator to develop your application.

Now that you've created a Quick Start Wizard application and tried some of the other Wizards, this chapter provides an overview of how to create an application *without* using Wizards. It documents how the Application Generator ties everything—the Clarion language and all the parts of the development environment—together.

TEMPLATE DRIVEN PROGRAMMING

Clarion's Application Generator is "template driven." This means that it is a tool that changes based on the needs of the template you are using to generate your code. Clarion's Procedure, Control, Extension, and Code Templates all write source code for you, giving you a tremendous productivity boost. Templates also provide many of the benefits of object oriented programming, especially reusability, without the overhead of learning an object oriented language.

In Clarion for Windows, a template is not a "one-time" tool, but is a continually interactive tool that asks specifically for the information it needs to generate your source code. Changing the information you provide the template results in different source code generated the next time you make the application.

All the templates are stored in the registry file (REGISTRY.TRF). This file contains the pre-written executable code and data structures which you customize and reuse.

You can modify the default Clarion templates. You may also add third party templates, or write your own, and use them *in addition to, and along with,* the defaults. This makes the Application Generator an infinitely extensible tool.

To use the templates, when you create a new procedure you identify the Procedure template that generates code closest to the task you need to perform, then customize it with the development tools. These Procedure templates include elements such as "browse windows" for viewing groups of records, and "form windows" for editing one record at a time. If the procedure is for a window with a menu, the menu actions are automatically added to the application's procedure tree, and marked "To Do," as any other procedure call would be.

The usual way to begin customizing a procedure is to call one of the formatters. The Window and Report Formatters are visual design tools that allow you to "point and click" to design windows and reports. You just pick a control from a toolbox, CLICK to place the control, then RIGHT-CLICK to modify its properties.

Once you have created a procedure you can also use the Control, Extension, and Code templates to add even more functionality to the procedure. Control templates contain controls and all the standard executable code needed to use and maintain them. All the pre-populated controls that appear in the default window designs for the Procedure templates are actually Control templates to perform all the functions of the procedure.

Extension templates add executable code that increase the functionality of the procedure. Each typically provides you with on screen instructions on what information to "fill in" to incorporate its functionality into the application.

Another way you may customize a procedure is to add embedded source code. The Application Generator displays a tree displaying all of the "embed points" that you have: before, during, and after the procedure, and for all the events the window or controls in the procedure generate. You can pick the precise logical point at which to execute the code, then "hand code" it, or use a Code template to write the code for you. The Application Generator generates all your application's source code from the templates and all your customizations (including embedded source code).

Clarion for Windows provides a rich assortment of standard templates with which you can rapidly develop applications. Just as the *Quick Start Tutorial* chapter introduced you to the Wizards, the rest of the *Getting Started* manual introduces you to using Clarion's Templates in the Application Generator to produce any Windows application you need.

A WALK THROUGH THE DEVELOPMENT ENVIRONMENT

Clarion's Development environment contains seven main functional parts, all of which are accessible from the others. When using the Application Generator, buttons in the various dialogs lead to the other parts. The Application Development Flow chart at the beginning of this chapter shows how the parts interact with each other and the template registry, with the Application Generator at the center of the whole process.

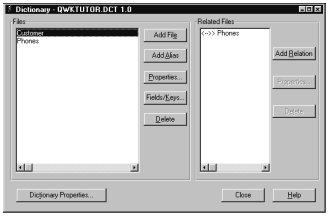
This section provides a description of each part, in the order that a typical programmer using the Application Generator might encounter them. Each contains dialog boxes which the programmer fills out to "describe" the Application's functionality to the Application Generator.

Programming Clarion for Windows is simply a "walk through" a series of dialog boxes. There's no mandatory sequence in which you must "fill in" the dialogs, though you do need to create some files before others. If you know which dialogs do what, it makes building your application that much quicker.

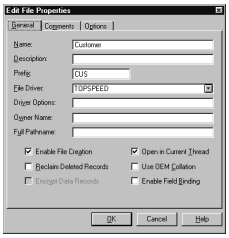
The Dictionary Editor

The Data Dictionary, maintained by the Dictionary Editor, holds a description of the database, including its files, fields, relations, field validation rules, and referential integrity constraints. It's the first file you create when you design your application.

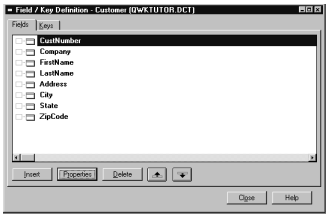
The other parts of the development environment use the options you set in the dictionary to let you easily place data fields in dialog boxes and reports you design. The Application Generator creates code to maintain the data files based on how you construct the Data Dictionary.



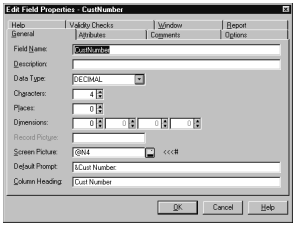
Start a new dictionary with the **File ➤ New** command, then select the *Dictionary* tab. This leads you to the **Dictionary** dialog. Define your application's data files and aliases in this dialog. It also shows the relationships between files. Buttons lead to the **New File Properties**, the **New File Alias**, and the **New Relationship** dialogs.



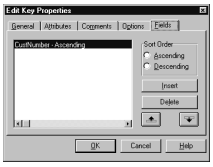
Specify the name and file driver for each data file, one by one, in the **New File Properties** dialog. It also allows you to set options such as Threaded, which specifies each execution thread accessing the file gets its own record buffer. This is useful for MDI applications.



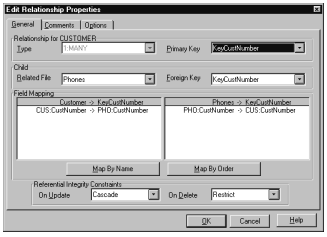
From the **Field/Keys Definition** dialog, press the **Insert** button to specify fields, keys, and index files.



Define fields, their data type and size in the **New Field Properties** dialog. You can even pre-define control properties such as text justification. Two tabs provide shortcuts to all the properties dialogs in the Window and Report Formatters.



Specify the key components in the Fields tab of the **Edit Key Properties** dialog. Clarion for Windows automatically builds the key correctly even if you specify multiple field types.



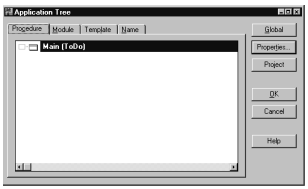
Define relationships in the **New Relationship Properties** dialog. You can also specify Referential Integrity Constraints from controls in this dialog. With the major parts of the dictionary defined, you save the dictionary and move on to the .APP file.

The Application Generator

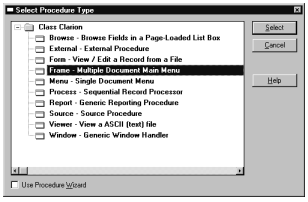
The Application Generator generates your application's source code, based on the Procedure templates you use. It allows you to add global and local memory variables, and customize the procedures with Control, Extension , and Code templates along with embedded source code.

The Application Generator also provides access to other parts of the IDE to customize the look and functionality of the windows, menus, reports and other user interface elements.

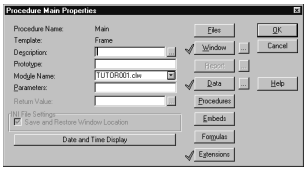
Start a new application with the **File ➤ New** command, then select **Application**. This allows you to set the basics—application name, data dictionary name, help file and the application template—in the **Application Properties** dialog. This creates the .APP file and displays the Application Tree.



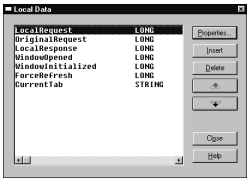
View and maintain the parts of your application in the **Application Tree** dialog. It hierarchically displays your application's procedures, and marks the ones still to be defined as "To Do." A button press accesses the **Select Procedure Type** dialog.



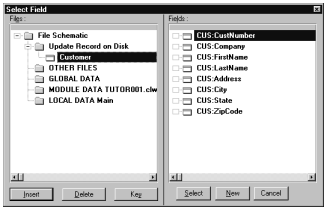
Select the Procedure template for a "To Do" procedure in the **Select Procedure Type** dialog. Procedure templates such as Browse and Form appear in a list. The **Select** button accesses the **Procedure Properties** dialog.



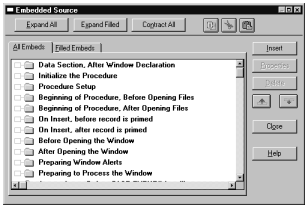
The **Procedure Properties** dialog is the hub for all the other dialogs that let you customize the procedure so that your application does the job the way you want. Press the **Data** button to define local memory variables.



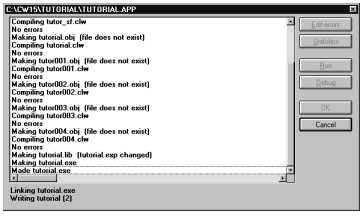
Define and set the order the program initializes local memory variables in the **Data** dialogs. Press the **Insert** button to define variable name, type, size etc., in the **New Field Properties** dialog.



Select the files, keys, aliases, and fields the procedure, the Control template, and/or the controls will access in the **Select Fields** dialog.



Press the **Embeds** button to display the **Embedded Source** dialog. This allows you insert custom code or Code templates at points before, during, and after the procedure, or on window and field-specific events. Select the embed point and press the **Add** button.

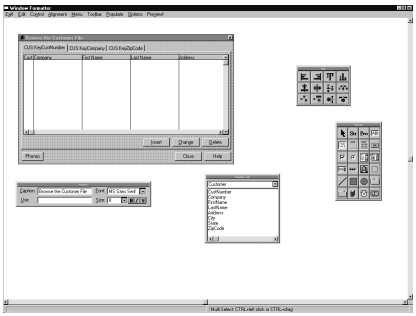


After you've customized the procedure using the Window Formatter, Report Formatter and/or Text Editor, you can back up to the Application Tree to generate and compile the code!

The Window Formatter

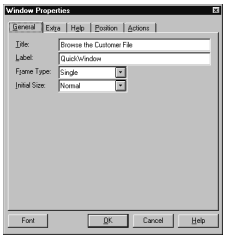
You visually design your application's windows and controls—everything the end user sees—in the Window Formatter. It automatically generates source code for the elements you visually design on screen.

Most likely, when using the Application Generator, you'll call the Window Formatter by pressing the **Window** button in a **Procedure Properties** dialog, to customize a window or dialog box.

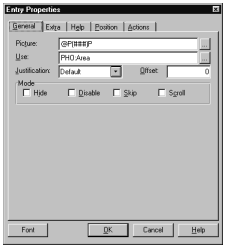


The Window Formatter provides a view of the window under development. CLICK in the toolbox, then CLICK in the window to place a new control in the window.

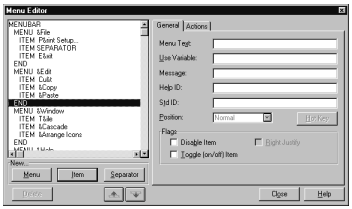
You can then press the **Preview!** menu item to see exactly how the window appears to the end user.



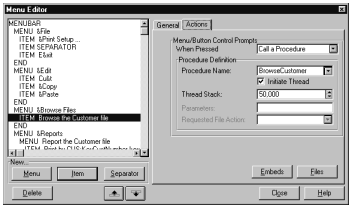
Each window and each control in the window has an associated property dialog that controls its appearance. Select the window or anything in it, then RIGHT-CLICK and choose **Properties** from the popup menu. The **Window Properties** dialog sets basic elements such as system menus, caption, etc.



A typical **control property** dialog sets options such as a label, its field equate label to reference it in executable code, or if an entry box, a field or variable name to reference its contents.

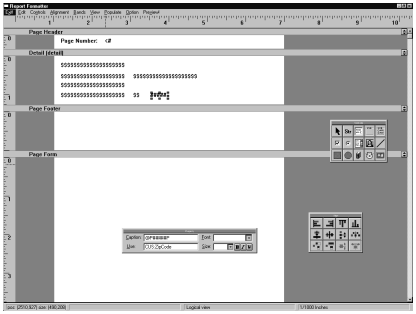


Choose **Menu ➤ Menu Editor** to call the Menu Editor (if the window has a menu). Edit the menu text, and new menu items with the **Item** button, and specify menu item functionality by pressing the **Actions** tab.



The **Actions** tab displays the prompts for execution options. For a menu item, you name a procedure to call, so that when the user selects the menu command, it executes the procedure you name.

The Report Formatter

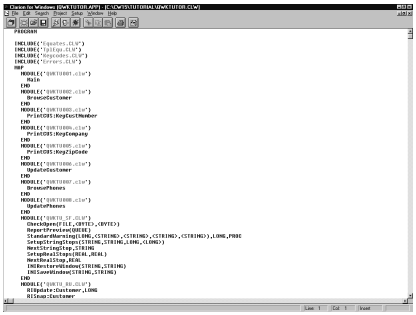


The Report Formatter works with the Application Generator in much the same way as the Window Formatter. You place controls in bands. At run time, the print engine processes the records you specify, printing the fields you specify in the Detail band.

The Text Editor

The Text Editor is a full function programmer's editor in which you can "hand-write" source code.

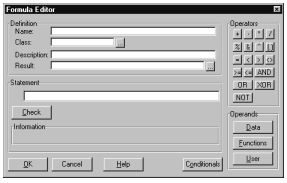
Most likely, when using the Application Generator, you'll call the Text Editor to create embedded executable source code to customize the way a procedure operates.



The Editor features color coded syntax highlighting, making it easier to identify the different parts of the Clarion language statements for editing purposes.

It also has full text search and replace capabilities, along with all the standard editing tools.

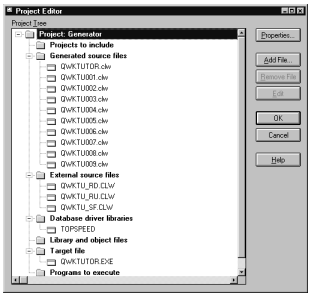
The Formula Editor



The Formula Editor helps you to quickly generate expressions that result in a value assigned to a variable. You can use the Formula Editor to create computed fields (simple assignments) and conditional fields (multiple, mutually exclusive, assignments based on some condition).

The Project System

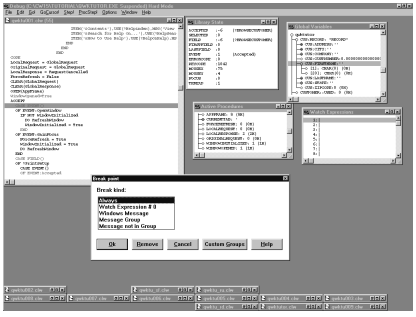
The Application Generator automatically creates the project for the application. The project contains compiler and link options, such as whether to include debug code, optimization choices, external drive files, and so on.



The Project Tree displays the source code files, libraries and other external files for the project. Press the **Properties** button to set specific options. Usually, when creating an application using the Application Generator, you'll just press the **Make** button to compile and link the application. The only Project system dialog you'll encounter is the **Compile Results** dialog, which allows access to the Debugger.

The Debugger

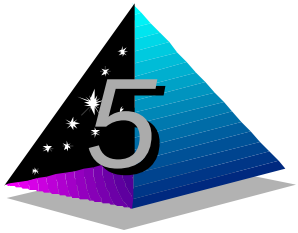
Debugging a program usually requires running the program and repeatedly stopping it to examine the value of different variables. The Debugger contains a number of windows which display source code, variable contents, active procedures and more.



To view a source code window, you tell the project system to include debug information in the .EXE file, then start the Debugger by pressing the **Debug** button in the **Compile Results** dialog, or choosing **Project ➤ Debug**. The simplest way to debug your application is to identify the part of the program that you think is producing the bug, and set a breakpoint, using the *Breakpoint* dialog, at that part of the code. You can then run the program, and the Debugger will suspend it at that point so that you can examine the value of the variables. This will help you pinpoint the error so that your application is perfect!

PLANNING THE APPLICATION

Contents



As a general rule, every minute you spend planning your application beforehand saves you ten later. This chapter informally describes the planning process for the application you'll create in the subsequent chapters. In the real world, you'll probably create a bona fide functional specification for your important applications. This informal description defines:

- The tasks the application performs.
- The data the application maintains, and how it stores it.
- How the user executes the application—its user interface.

As a basis for the application, this *hands on* tutorial application uses the data dictionary from the application you just created using the Quick Start Wizard. It extends the concept to a simple Order/Entry system, using the data dictionary for keeping track of customers.

APPLICATION TASKS

This application maintains the customer and billing files for the *Wibbly Wobbly Widget* manufacturing company. The first task in planning just what the application is to do is to assess what the company expects it to do.

For the purposes of the tutorial, the application is a simple order/entry system. Customers typically phone in orders for one or possibly two different types of Widgets at a time. A salesperson takes the order. The billing department prints out an invoice that night. When the customer sends in payment, the accounting department posts the checks.

The application therefore must provide:

- Entry dialogs for taking the order, or modifying the data in it later.
- Access to the customer list from within an order entry dialog. The customer list is the one you created with the Quick Start Wizard, stored in the Customer file.

- Access to the list of widget part numbers (items) that Wibbly Wobbly manufactures, from an order entry dialog.
- Browse windows for listing sales transactions.
- Entry dialogs to maintain the items list.
- Printed reports.

THE DATA AND THE DATA FILES

The first task in planning the file structure is to assess what data the application needs, and how to store it with the minimum amount of duplication.

Good database management maintains separate data files for each "entity" or group of discrete data elements. The data "entities" this application maintains are:

Customer	Customer name and address data that changes only when a customer moves. Created in the Quick Start tutorial, along with its related Phones file.
Order	Basic information needed for assembling the data needed to print an invoice. It "looks up" information from the other files, such as the customer name and address. When a sales person takes a new order, they add a record to this file.
Detail	Product, price, and quantity ordered for an item on a given invoice: the variable information for each order. Though this duplicates price information in the Products file, you must maintain the price at the time of the sale here. Otherwise, when you increase the price in the Products file, it would cause the balance in the Detail file to change.
Products	Information on the products sold by the company, including product number, description and price. This data changes only when a price changes or a new product is added.

The Customer File

The Customer file stores "constant" data such as customer names and addresses. It's most efficient to store this data in one place, allowing for a single update when the information changes. This also saves space by eliminating redundant customer information in the Order file; otherwise, if there were 1000 orders for company XYZ, the address information would be repeated 1000 times. Reducing storage requirements by storing the data only once is called *normalization*.

The customer data requires a field to uniquely identify the customer. The company name is unsuitable because there could be duplicates. There may be, for example, multiple records for a customer called "Widget Depot," if it has multiple locations. The Quick Start application *already* specified that the CustNumber field is an auto-number key which automatically creates and stores unique customer numbers.

The CustNumber field serves as the *primary key* for the data file. Any other data files which are related to the Customer file must declare the CustNumber field as a *foreign key*. A *primary* key is a field, or combination of fields, that uniquely identifies each record in a data file. A *foreign* key is a field, or combination of fields, in one file (or table) whose key data *must* match a *primary* key record in another related file.

Because there may be many orders for each customer number, the relationship between the Customer file and the Order file will be a *one to many* (1:Many) relationship. We say the customer data file is the *parent* file, and the Order data file is the *child* file.

The Phones File

The Phones file stores telephone numbers—each customer could have several. Each record includes a CustNumber field to relate back to the Customer file.

The Phones file also includes a text field in which we can indicate whether the phone number is an office, fax, mobile or home number. Using the data dictionary, we'll specify that the control for entering data for this field should be a drop-down list with the choices already loaded.

The Order File

The Order data file gathers information for each sales transaction from all the other data files (such as the customer data). Because much of the basic data in this file prints in the "header" area of the invoice, this is sometimes called the Order Header.

Every sales transaction requires one record in the Order file. The record relates to the customer information by referencing the unique customer number. Because some order records may reference one product, and others may reference ten, you'll create a separate Detail file which relates back to a unique order number. This creates a one to many relationship, with the Order file as parent and Detail as child. The actual products ordered are identified by their product codes, in the Detail file.

The Order record thus holds a customer number to relate back to the customer data (the foreign key), and a unique Order number to relate to the Detail. You'll create a multi-component primary key on the two fields, so that you can easily create a browse sorted by customer and invoice number.

The Detail File

The Detail file stores the products ordered by their product codes (a foreign key into the Product file), their individual prices, the quantity of each, and the tax rate. An additional field holds an invoice number, which relates back to the Order file in a *many to one* relationship.

The Detail file duplicates the price information with the fields in the product file; this is because prices may change. It's important to store the price field within the detail file record because if the price increases in six months, today's paid in full invoice would reflect a balance due.

The Product File

The Product file stores unique product numbers, descriptions, and prices. When the sales person looks up a product by name, the application inserts the product number into the Detail record. The product code is the primary key—no two items can have the same code, and every product must have a code. An additional field contains the tax rate for the product.

Referential Integrity

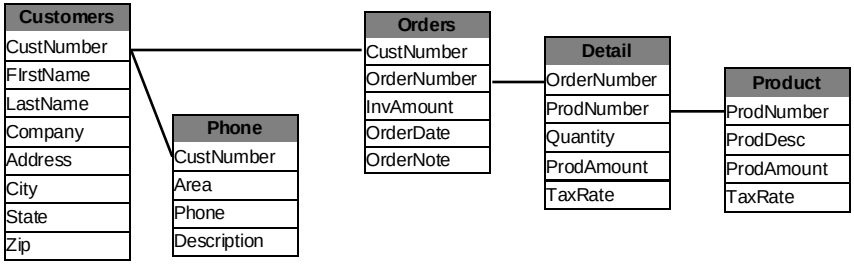
Referential Integrity refers to the process of checking all updates to the key field in a given file, to ensure that the validity of parent-child relationships is correctly maintained. It also refers to ensuring that all child file records always have associated parent records so that there are no "orphan" records in the database.

Because the data for a given transaction resides across several files, this application must enforce referential integrity. This is critical, yet many database application development tools require you to hand code procedures to enforce this. The Application Generator's templates implement this automatically in your generated source code when you select a few options in the Data Dictionary.

It is essential that the application not allow an update to a record that leaves a blank or duplicate value in a primary key field. For example, we need to restrict the ability of the end user to update a record in a way that could cause a duplicate Customer number. If two different companies shared a duplicate Customer number, you could send a bill to the wrong company.

The Complete Database Schematic

The schematic below provides an overview of the entire database. If you look at it from the point of view of the sales agent taking a phone order, the Order file records who's ordering, the Detail stores what they're ordering, and the Customer and Product files store constant information about the customers and products.



The item code looks up the description and price. The customer code looks up the customer's name and address. Other data, such as the transaction date, fill in automatically (by looking up the system date, for example).

Finally, the tutorial will create a brand new data dictionary, and you will copy and paste the files that Quick Start defined for you into the new dictionary.

As for the actual application you create, because the tutorial is a teaching tool more concerned with showing what Clarion for Windows can do for you, it won't create a full-scale order entry system. However, you will find that some parts of the application will be very "showy," so that you can quickly learn how to do equivalent procedures in your applications.

APPLICATION INTERFACE

The next major task before coding is to plan the user interface. For a business application like this, it's crucial that a salesperson quickly locate the data they need quickly, so that they can record the sale and move on to the next phone call. Therefore, the application should put all the important data "up front" by default, and no entry or maintenance dialog should be more than a button or menu command away.

Additionally, the company uses many other Windows applications; therefore, it's especially important that the application have a standard windows "look and feel." End users learn a familiar interface more quickly.

To implement the tasks the application must execute in a consistent manner with our guidelines, we can plan for the items listed below. Though the following is no substitute for a real program spec, it should suit us for Tutorial purposes.

- Because the application will handle the maintenance for the customer, item, and billings files on different forms, the Multiple Document Interface (MDI) is necessary.
- The application should have a toolbar with buttons to load forms and browse windows.
- To maintain a consistent "look and feel," the main menu commands will be File, Edit, Insert, View, Window, and Help. The file menu accesses the printing and exit procedures. The Edit menu calls the form dialogs for editing a current record (if highlighted in a browse). This duplicates the buttons in a standard browse procedure, but we include it because it's a standard Windows app menu. The Insert menu adds (or inserts) a new record in a data file. The View menu contains the procedures for browsing a data file. Window and Help perform standard actions.
- When adding new orders, the sales people should be able to pick customers and products from scrolling lists. Pertinent data in the order dialog—addresses, item descriptions and prices—should automatically "fill in."

Now that the application description is complete, we're ready to begin work. The first step is to create the data dictionary.

CREATING A DATA DICTIONARY

This chapter teaches you how to:

- Create a new data dictionary.
- Copy and customize file definitions from the Quick Start data dictionary to the new one.
- Relate the files and specify Referential Integrity constraints.
- Pre-format window controls for the fields.

This tutorial assumes that you've completed the *Quick Start Tutorial* in chapter three.

Tutorial Files

We recommend that you follow through the entire tutorial, especially if the Clarion development environment is brand new to you. Clarion's Application Generator approach to programming is sufficiently different from other approaches to programming that we hope you'll thoroughly immerse yourself in order to get the most out of your new tool. If you're already an experienced Clarion programmer, you may prefer to just examine the completed tutorial files rather than step through the tutorial. The completed tutorial files reside in the \CW15\EXAMPLES\TUTOR directory.

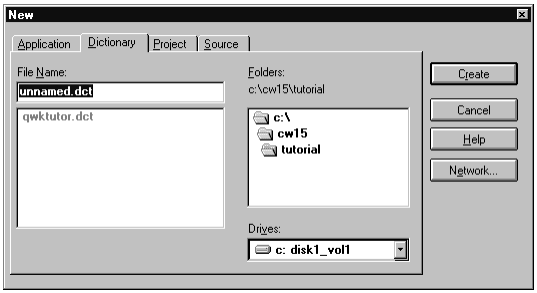
CREATING THE DICTIONARY

Whenever you create a new application, you first define the data dictionary (.DCT file). From the data dictionary, the Application Generator obtains all its information about the data files your application uses, their relationships to one another, plus additional information such as predefined formatting for controls.

When you ran the Quick Start, though you didn't actually use the Dictionary Editor, you first defined the data dictionary. This introduces the Dictionary Editor.

❏ Fill in the name of the new data dictionary.

1. Choose **File ➤ New** from the menu.
2. Select the **Dictionary** tab then press the **Create** button.



The **Dictionary** dialog appears. The caption bar indicates its name is UNNAMED.DCT.

❏ Name and save a new dictionary file.

1. Choose **File ➤ Save As** from the menu.

   The **Save Dictionary** dialog appears.
2. Select the subdirectory (for example, TUTORIAL, below the Clarion for Windows directory) in the **Directories** list.
3. Type *TUTORIAL* in the **File Name** field.

   Clarion for Windows appends the file extension; TUTORIAL.DCT is the full name for the dictionary file.
4. Press the **OK** button to save the file.

   This saves only an empty dictionary file. The caption bar now shows the file name.

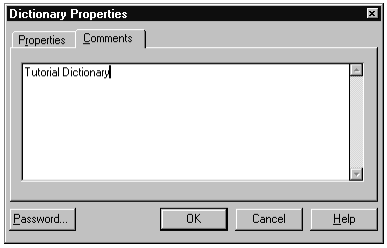
❏ Specify a description for the dictionary:

1. Press the **Dictionary Properties** button.

   The **Dictionary Properties** dialog appears.
2. Select the **Comments** tab then type *Tutorial Dictionary* in the text field.

The **Comments** tab allows you to write free form text notes regarding the dictionary. It's optional, but extremely useful for programmers who may have to return to a project for maintenance after an interval of months.

This dialog also provides a **Password** button, which allows you to prevent others from using this dictionary. There's no need to fill it in for the tutorial, but it's a useful feature to keep in mind.



3. Close the **Dictionary Properties** dialog by pressing the **OK** button.

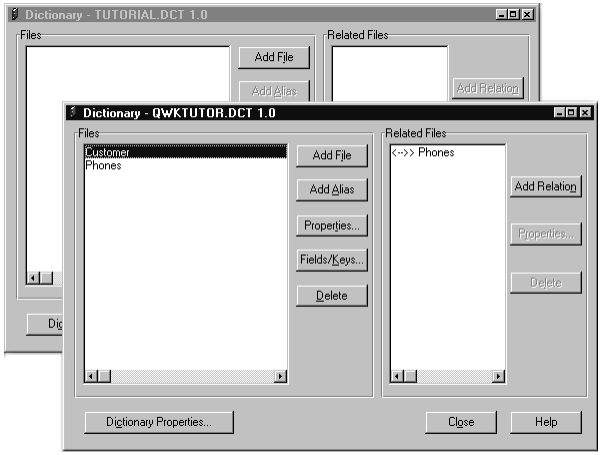
COPYING FILE DEFINITIONS FROM ONE DICTIONARY TO ANOTHER

You can use the copy and paste commands to copy file definitions from another dictionary.

❑ Open the other data dictionary, select a file, and copy:

1. Choose **File ➤ Pick** from the menu then select the **Dictionary** tab.
2. Select the *QWKTUTOR.DCT* file from the file list, and press the **Select** button.

   Another Dictionary dialog opens, containing the file definitions from the Quick Start application.



3. Select the **Customer** file from the **Files** list.
4. Choose **Edit ➤ Copy**.
5. Press CTRL+F6, or CLICK on the *TUTORIAL.DCT* **Dictionary** dialog.

   This makes the Tutorial dictionary the active dictionary.
6. Choose **Edit ➤ Paste**.

   The **Edit File Properties** dialog appears.

   Not only does this operation copy the file definition; it copies the keys as well.
7. Press the **OK** button to close the **Edit File Properties** dialog.

Copy the Phone File Definition

Now use the copy and paste commands to copy the other file definition.

❏ Select a file, and copy:

1. Press CTRL+F6, or CLICK on the *QWKTUTOR.DCT* **Dictionary** dialog.
2. Select the **Phones** file from the **Files** list.
3. Choose **Edit ➤ Copy**.
4. Press CTRL+F6, or click on the *TUTORIAL.DCT* **Dictionary** dialog.

   This makes the Tutorial dictionary the active dictionary.
5. Choose **Edit ➤ Paste**.

   The **Edit File Properties** dialog appears.
6. Press the **OK** button to close the **Edit File Properties** dialog.

❏ Close the Quick Start dictionary file.

1. Press CTRL+F6, or CLICK on the *QWKTUTOR.DCT* **Dictionary** dialog.
2. Press the **Close** button, or choose **File ➤ Close**.

RELATING THE FILES AND SETTING REFERENTIAL INTEGRITY OPTIONS

You can copy file definitions (including their keys), but Clarion for Windows does *not* allow you to copy relations from other dictionaries. Therefore, you must define a relationship for the two files.

❏ Define the first side of the relation.

1. Highlight the *Phones* file then press the **Add Relation** button.

The **New Relationship Properties** dialog appears. Because the last file selected was the *Phones* file, we'll set up the relationship from its perspective.

Each Customer can have more than one phone. Customer is the parent, in a *parent-child relationship.* Therefore, it's a **MANY:1** relationship, from Phones into Customer.

2. Choose **MANY:1** from the **Type** dropdown list in the **Relationship for Phones** group box.
3. Choose *KeyCustNumber* from the **Foreign Key** dropdown list.

This is the key that matches a primary key in the Customers file; therefore, it's a foreign key.

❑ Define the other side of the relation.

1. Choose *Customer* from the **Related File** dropdown list in the **Parent** group box.
2. Choose *KeyCustNumber* from the **Primary Key** dropdown list.

In the *parent-child relationship* the foreign key in the child must relate to a primary key in the parent. See the *Using the Data Dictionary* chapter in the *User's Guide* for a brief discussion of relational database theory.

3. Press the **Map by Name** button to link the fields.

Set Referential Integrity Constraints

By setting Referential Integrity constraints, you can specify how the Application Generator writes the source code that handles what happens if an end user attempts to modify a value in a primary key, or attempts to delete a parent that has children. If you don't set constraints, an end user can compromise the integrity of the database by creating "orphan" records one of two ways: by deleting a parent with children, or changing the parent's linking field value.

For the tutorial, specify that the application should *update* the foreign key record, if the primary key field value is changed. Also, specify that it should *not allow* the user to delete a parent with children.

1. Choose **Cascade** from the **On Update** dropdown list in the **Referential Integrity Constraints** group box.

Cascading a change means that the application extends the change and updates the foreign key (child file) field, for all the child file records related to that one parent file record.

2. Choose **Restrict** from the **On Delete** dropdown list.

   Restricting a delete means that the application does not allow deleting a parent with children.

3. Press **OK** to close the **New Relationship Properties** dialog.

At this point, the *top* of your **Dictionary** dialog looks like this:

Look at the small arrows to the left of the related file name in the **Related Files** list. It indicates the nature of the relationship between the two files. Two angle brackets (>> or <<) point to the many side from the one side. One angle bracket (> or <) points to the one side from the many side.

Therefore, **Phones <<-> Customers** indicates *Many* phones file records can be related to *One* customer file record.

Save Your Work

It is a good habit to save your work frequently while developing your applications. To do so, choose **File ➤ Save**, or press the *Save button* on the tool bar. This writes the dictionary file to disk.

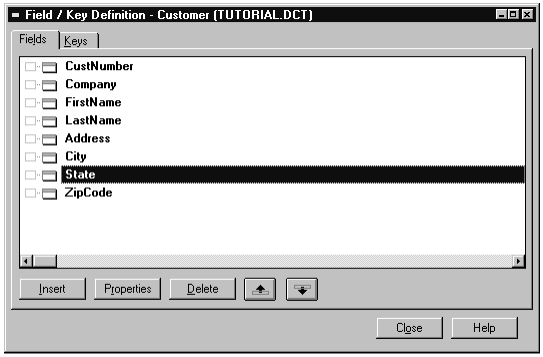
PRE-DEFINING WINDOW CONTROL FORMATS

Within the data dictionary, you can specify the default properties of the window controls which reference the fields you define. You can also specify certain Data Integrity rules by setting Validity Checks and the field's initial value.

❑ Access Field Properties:

1. Highlight the *Customer* file in the **Files** list.
2. Press the **Fields/Keys** button.

   The **Field/Key Definition** dialog allows you to edit the properties for any field or key in the file.



3. Select the *State* field, and press the **Properties** button.

   The **Edit Field Properties** dialog appears, showing the options Quick Start filled in for this field.

❑ Set Validity Checks in preparation for creating a drop down list box for the State field:

1. Select the **Validity Checks** tab.

   The **Validity Checks** tab allows you to set numeric ranges for number fields, specify that a field value must match another field in a related value, must be true or false, and in this case, that the field value must be in a list you specify in this dialog.

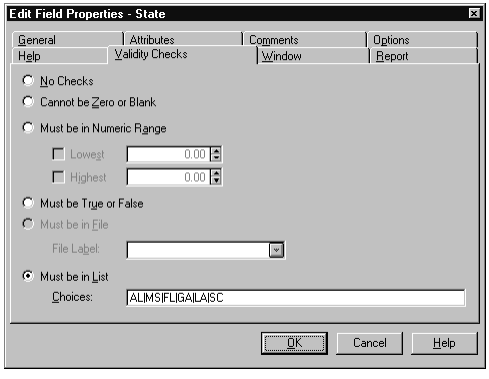
2. Select the **Must be in List** radio button.

3. Type the following in the **Choices** box:

   ```
   AL|MS|FL|GA|LA|SC
   ```

   A vertical bar (|) must separate each choice.

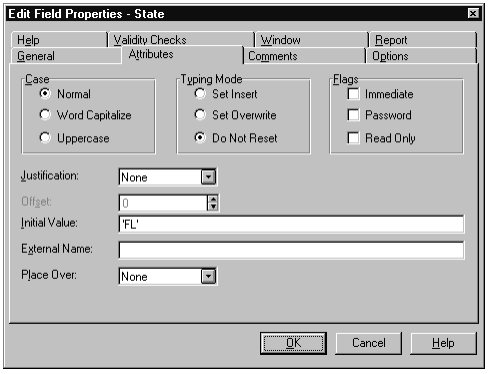
   This defines the actual list of allowable choices. In this case, the dictionary specifies that only the state abbreviations for these six southern states is acceptable. You will specify that the default control for this field is a drop down list.



❑ Set a default value for the state field:

1. Select the **Attributes** tab.
2. Type *'FL'* in the **Initial Value** field (including the single-quote marks).

   This specifies that anytime the control appears, its default value will be "FL." Initial values can be time savers for the end user; in this case, if most customers were located in "FL," it saves picking it from the list each time a new customer has to be added. The single-quote marks are necessary because you can also name a variable or function as the initial value. In this case, the initial value is a string constant.



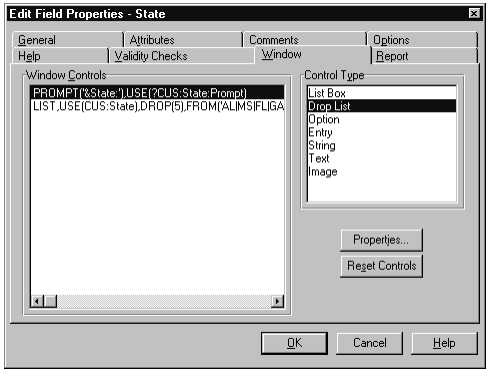
❑ Specify that the default window control for the state field should be a dropdown list.

1. Select the **Window** tab.

   When you specify a **Must be in List** option, the default window control for the field is an OPTION structure with RADIO buttons. These appear by default in the **Window Controls** list.

2. Select **Drop List** from the **Control Type** list box.

   The **Window Controls** list now updates to show only a PROMPT and a LIST control with a DROP attribute.

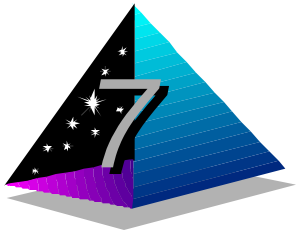


3. Press the **OK** button to close the **Edit Field Properties** dialog.
4. Press the **Close** button to close the **Field/Key Definition** dialog.
5. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

In the next chapter, you'll learn how to add a file to the data dictionary, starting totally "from scratch."

ADDING FILES AND FIELDS

Contents



After copying and modifying the two files defined in the Quick Start application, you're ready to add a new file from scratch. At this point, only the **Data Dictionary** dialog for the Tutorial dictionary should be open.

DEFINING NEW DATA FILES

❑ Specify the label, prefix, and description for the data file.

1. Press the **Add File** button in the **Dictionary** dialog.
2. When the **Add File** dialog appears asking whether you wish to use Quick Load, press the **No** button.

   The **New File Properties** dialog appears. Clarion uses the information you fill in here for the FILE data structure declaration.
3. Type *Orders* in the **Name** field, then press TAB.

   The **Name** field only accepts a valid Clarion label, which uniquely identifies the data structure. A label may only contain letters, numbers, and the underscore(_) or colon (:) characters, and must begin with a letter or underscore. Executable code statements use this label to refer to the file.

   After pressing TAB, "ORD" automatically appears in the **Prefix** field. The prefix is one way to to uniquely identify fields of the same name in different data files. For example, *ORD:CustNumber* is the *CustNumber* field in the *Orders* file while *CUS:CustNumber* is the *CustNumber* field in the *Customers* file. You can also uniquely identify fields by using Field Qualification syntax (discussed in the *Language Reference).*
4. Type *Order header file* in the **Description** field.

   This description appears next to the data file label in the *Dictionary* dialog list. If you press the **Comments** tab, you can type a long text description. A description of what the file was designed for, and why you designed it, can be very helpful for when you "return" to the file for maintenance programming.

❑ Choose the file driver.

1. Choose **TOPSPEED** from the **File Driver** dropdown list.

   This declares the file format for the data file as the TopSpeed file format.

   The *Database Drivers* appendix in the *User's Guide* provides information about what data types each driver supports, plus other useful information such as the default file extensions for data and/or index files. It also provides tips and tricks for choosing the right driver for the job, such as which drivers are best when your application must handle a very large database which is frequently updated, or which drivers are best when the quickest query time is the foremost concern.

2. Press the **OK** button.

   You can accept the defaults for all other options in the dialog. The dialog box closes, and the **Dictionary** dialog lists the Orders file, with "Order header file" listed next to it.

Name the Detail and Products Data Files

1. Press the **Add File** button in the **Dictionary** dialog.

   Choose **No** when asked if you wish to use Quick Load.

2. Type *Detail* in the **Name** field.
3. Type *Order Detail file* in the **Description** field.
4. Type *DTL* in the **Prefix** field.

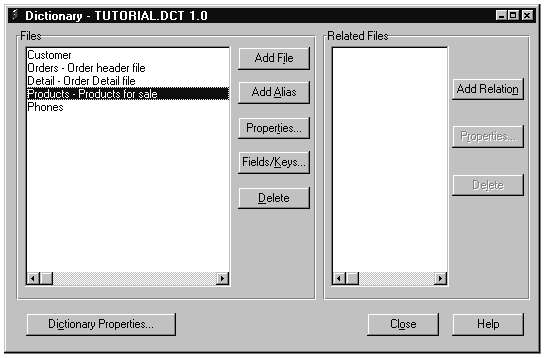
   By customizing the default prefix (changing it from "DET" TO "DTL"), you can make your code more readable. Three characters is the convention for file prefixes, but you are not limited to that.

5. Choose **TOPSPEED** from the **File Driver** dropdown list.

   Accept the defaults for all other options in the dialog.

6. Press the **OK** button.
7. Press the **Add File** button (don't use Quick Load).
8. Type *Products* in the **Name** field.
9. Type *Products for sale* in the **Description** field.
10. Type *PRD* in the **Prefix** field.
11. Choose **TOPSPEED** from the **File Driver** dropdown list.
12. Press the **OK** button.
13. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

At this point, your Dictionary dialog looks like this:



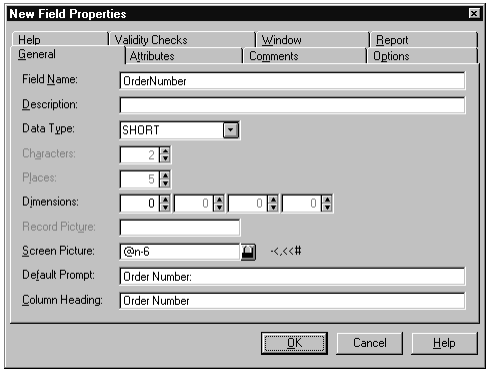
DEFINING FIELDS FOR THE ORDERS FILE

At this point, go back to the Orders data file and prepare to define its fields.

❑ Open the **New Field Properties** dialogs.

1. Highlight the **Orders** file in the **Files** list.
2. Press the **Fields/Keys...** button.
3. Press the **Insert** button to open the **New Field Properties** dialog.

Once you begin the process of defining new fields, an empty **New Properties** dialog automatically appears after you add each successive field. This speeds up the process of adding multiple fields. After adding your last field, you just have to press the **Cancel** button on the blank dialog to return to the **Field / Key Definition** dialog.



❏ Define the CustNumber field to match the field in the Customers file.

This allows you to relate the files later.

1. Type *CustNumber* in the **Name** field.
2. Choose **DECIMAL** from the **Type** dropdown list.

   This matches the default data type used by the Quick Start Wizard.
3. Type *4* in the Characters field.

   This specifies the number of places left of the decimal point.
4. Press the **OK** button.

❏ Define the OrderNumber field.

This provides a unique identifier for each order

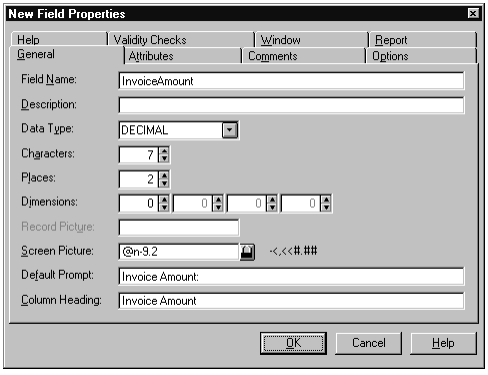
1. Type *OrderNumber* in the **Name** field.
2. Choose **SHORT** from the **Type** dropdown list.

   This specifies a short integer.
3. Press the **OK** button.

❏ Define the InvoiceAmount field.

This stores the total amount of the order.

1. Type *InvoiceAmount* in the **Name** field.
2. Choose **DECIMAL** from the **Type** dropdown list.



3. Type *7* in the **Chars** field.

   This specifies the total number of digits in the number (on both sides of the decimal point).
4. Type *2* in the **Places** field.

   This specifies the number of digits to the right of the decimal point.

5. Press the **OK** button.

❏ Define the OrderDate field.

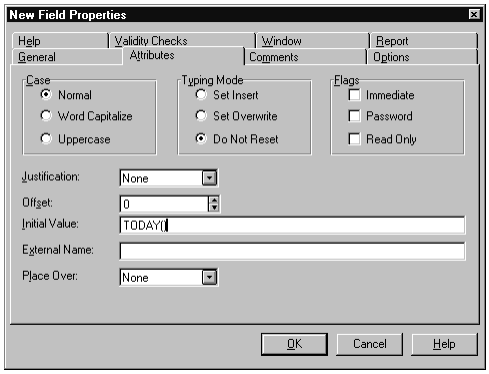
This stores the date the order was placed.

1. Type *OrderDate* in the **Name** field.
2. Choose **LONG** from the **Type** dropdown list.

   This is the preferred *date* storage data type for the TopSpeed driver.
3. Type *@d1* in the **Screen Picture** field.

   The screen picture specifies the default "character" formatting for a field. In this case @d1 signifies MM/DD/YY format. The dialog box displays a representation of the formatting next to the field.
4. Select the *Attributes* tab then type *TODAY()* in the **Initial Value** field.

   The generated source code places today's date in any control allowing a new record entry, using the built-in TODAY() function.



5. Press the **OK** button.

❏ Define the OrderNote field.

This allows for a short note for special handling instructions.

1. Type *OrderNote* in the **Name** field.
2. Choose **STRING** from the **Type** dropdown list.
3. Type *80* in the **Chars** field.

   This specifies 80 characters.
4. Press the **OK** button.

❏ Close the **New Field Properties** dialog and save your work.

All the fields are defined, and a blank **New Field Properties** dialog should be active.

1. Press the **Cancel** button to close the **New Field Properties** dialog.
2. Press the **Close** button to close the **Field / Key Definition** dialog.
3. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

DEFINING FIELDS FOR THE DETAIL FILE

At this point, go back to the Detail data file and prepare to define its fields.

❏ Open the **New Field Properties** dialogs.

1. Highlight **Detail** in the **Files** list.
2. Press the **Fields/Keys...** button.
3. Press the **Insert** button to open the **New Field Properties** dialog.

❏ Define the OrderNumber field.

This allows you to relate this file to the Orders file.

1. Type *OrderNumber* in the **Name** field.
2. Choose **SHORT** from the **Type** dropdown list.
3. Press the **OK** button.

❏ Define the ProdNumber field.

This allows you to relate this file and the Products file.

1. Type *ProdNumber* in the **Name** field.
2. Choose **SHORT** from the **Type** dropdown list.
3. Press the **OK** button.

❏ Define the Quantity field.

This stores the number of each product ordered.

1. Type *Quantity* in the **Name** field.
2. Choose **SHORT** from the **Type** dropdown list.
3. Press the **OK** button.

❏ Define the ProdAmount field.

This stores the unit cost of the product at the time of the order.

1. Type *ProdAmount* in the **Name** field.
2. Choose **DECIMAL** from the **Type** dropdown list.

3. Type *5* in the **Chars** field.
4. Type *2* in the **Places** field.
5. Press the **OK** button.

❑ Define the TaxRate field.

1. Type *TaxRate* in the **Name** field.
2. Choose **DECIMAL** from the **Type** dropdown list.
3. Type *2* in the **Chars** field.
4. Type *2* in the **Places** field.
5. Press the **OK** button.

❑ Close the **New Field Properties** dialog and save your work.

All the fields are defined, and a blank **New Field Properties** dialog should be active.

1. Press the **Cancel** button to close the **New Field Properties** dialog.
2. Press the **Close** button to close the **Field / Key Definition** dialog.
3. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

DEFINING FIELDS FOR THE PRODUCT FILE

At this point, go back to the Product data file and prepare to define its fields. This is the last file.

❑ Open the **New Field Properties** dialogs.

1. Highlight **Product** in the **Files** list.
2. Press the **Fields/Keys...** button.
3. Press the **Insert** button to open the **New Field Properties** dialog.

❑ Define the ProdNumber field.

This allows you to relate this file and the Detail file.

1. Type *ProdNumber* in the **Name** field.
2. Choose **SHORT** from the **Type** dropdown list.
3. Press the **OK** button.

❑ Define the ProdDesc field.

This allows for a product description

1. Type *ProdDesc* in the **Name** field.
2. Choose **STRING** from the **Type** dropdown list.
3. Type *25* in the **Chars** field.
4. Press the **OK** button.

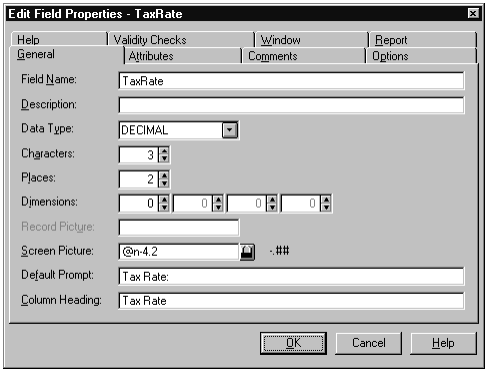
❏ Define the ProdAmount field.

This stores the unit cost of the product.

1. Type *ProdAmount* in the **Name** field.
2. Choose **DECIMAL** from the **Type** dropdown list.
3. Type *5* in the **Chars** field.
4. Type *2* in the **Places** field.
5. Press the **OK** button.

❏ Define the TaxRate field.

1. Type *TaxRate* in the **Name** field.
2. Choose **DECIMAL** from the **Type** dropdown list.
3. Type *2* in the **Chars** field.
4. Type *2* in the **Places** field.



5. Press the **OK** button.

❏ Close the **New Field Properties** dialog and save your work.

All the fields are defined, and a blank **New Field Properties** dialog should be active.

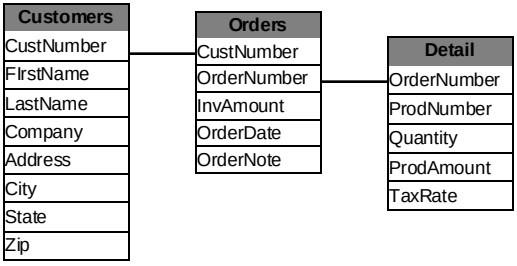
1. Press the **Cancel** button to close the **New Field Properties** dialog.
2. Press the **Close** button to close the **Field / Key Definition** dialog.
3. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

ADDING KEYS

Now that all the files are defined, we can add keys. The Quick Start Wizard already defined the keys for the two files you created for the QwkTutor application. In this chapter, we'll define keys for the remaining files.

DEFINING KEYS FOR THE ORDERS FILE

The fields in the Orders file that relate to other files in the database are the OrderNumber and CustNumber fields.



- The OrderNumber field relates to the Detail file.

  There should be no duplicate or null order numbers in the Orders file; this is a *primary* key.

  There may be more than one Detail record for a single matching Order Number. Therefore, this is a *One to Many* relationship, with the Orders file the "Parent" of the Details file.

- The CustNumber field relates to the Customers file.

  There will be duplicate values in the CustNumber field that relate to records in the Customers file. The key we define in the Orders file is a *foreign* key. The Customers file key does not allow duplicates and nulls, and was defined as the primary key for that file.

  More than one order record can exist for each customer, Making this a *Many to One* relationship, with the Orders file the "Child" of the Customers file.

Create the Primary Key

Make sure the Dictionary dialog for the Tutorial data dictionary is active.

❏ Name the Key

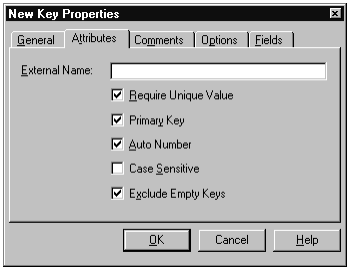
1. Highlight the *Orders* file in the **Files** list.
2. Press the **Field/Keys...** button.
3. Select the **Keys** tab.
4. Press the **Insert** button.

   The **New Key Poperties** dialog appears.
5. Type *KeyOrderNumber* in the **Key Name** field.

   As an easy naming convention, we suggest incorporating both the word “key” and the field name in the key name (just as the Quick Start Wizard does)
6. Select the **Attributes** tab, check the **Require Unique Value** box, then check the **Primary Key** box.

   This specifies the key is a primary key. The generated source code automatically prevents the end user from inserting duplicate or null values.
7. Check the **Auto Number** box.

   The generated source code increments the key field with each new record.



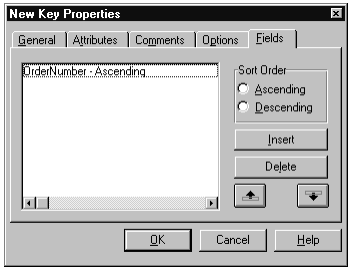
8. Select the **Fields** tab.

❏ Specify the key field

1. Press the **Insert** button.

   The **Insert Key Component** dialog appears, ready for you to specify a field or fields for the key.
2. DOUBLE-CLICK on *OrderNumber*.

   This adds the field to the list of component fields for this key.



2. Press the **OK** button.

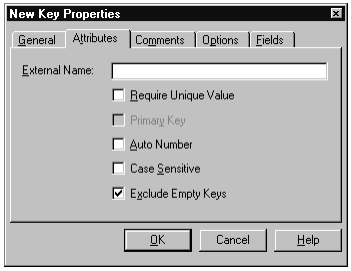
   A blank **New Key Properties** dialog appears, ready for you to specify another key.

Define a Foreign Key

Now you can define the CustNumber key. There may be duplicates within this file. It relates to the primary key in the Customers file, so therefore, this is a foreign key.

1. Type *KeyCustNumber* in the **Name** field.
2. Select the **Attributes** tab.

   The key *does* allow duplicates so leave all the default settings.

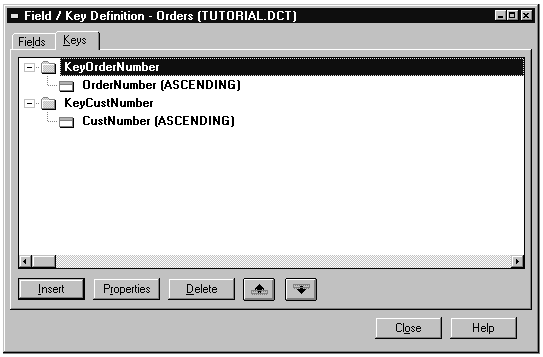


3. Select the **Fields** tab.
4. Press the **Insert** button.

   The **Insert Key Component** dialog appears, ready for you to specify a field or fields for the key.
5. Select *CustNumber* then press the **Select** button.
6. Press the **OK** button.

   Each time you finish a new key, a blank **New Key Properties** dialog appears, ready for the next.
7. Press the **Cancel** button to close the blank **New Key Properties** dialog.

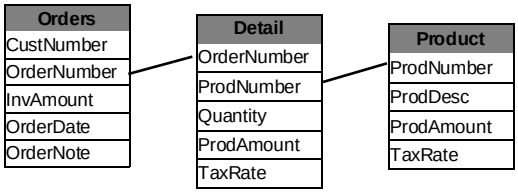
The **Field/Keys Definition** dialog for the Orders file now looks like this:



8. Press the **Close** button to close the **Field/Key Definition** dialog.
9. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

DEFINING KEYS FOR THE DETAIL FILE

The fields in the Detail file that relate to other files in the database are the ProdNumber and OrderNumber fields.



- The OrderNumber field relates to the Orders file.

  There will be duplicate values in the OrderNumber field that relate to records in the Orders file. The key we define in the Detail file is another *foreign* key. The Orders file key does not allow duplicates and nulls, and was defined as a primary key.

  There may be more than one Detail record for a single matching Order Number. Therefore, this is a *Many to One* relationship, with the Detail file the "Child" of the Orders file.

- The ProdNumber field relates to the Product file.

  There will be duplicate values in the ProdNumber field for the records in the Detail file. There may be more than one Detail record containing a single Product Number. Therefore, this is another *Many*

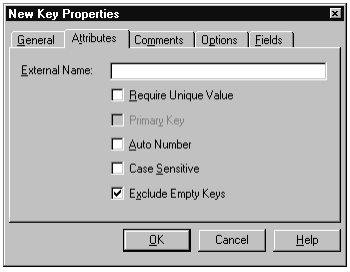
to One relationship, with the Detail file the “Child” of the Product file.

Define the First Foreign Key

Define KeyProdNumber so that there may be duplicate ProdNumber values in this file

1. Highlight the *Detail* file in the **Files** list.
2. Press the **Field/Keys...** button.
3. Select the **Keys** tab.
4. Press the **Insert** button.
5. Type *KeyProdNumber* in the **Name** field.
6. Select the **Attributes** tab.

   The key *does* allow duplicates so leave all the default settings.



7. Select the **Fields** tab.
8. Press the **Insert** button.
9. Select *ProdNumber* then press the **Select** button.
10. Press the **OK** button.

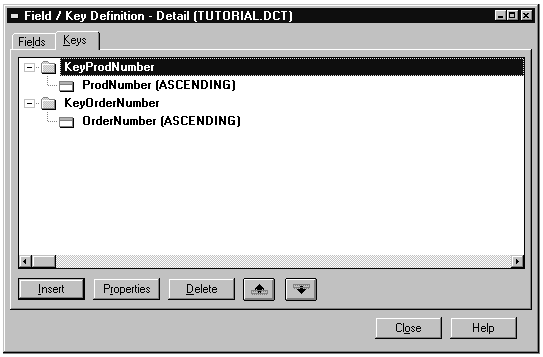
    A blank **Key Properties** dialog appears, ready for you to specify another key.

Define the Second Foreign Key

1. Type *KeyOrderNumber* in the **Name** field.
2. Select the **Fields** tab.
3. Press the **Insert** button.
4. Select *OrderNumber* then press the **Select** button.
5. Press the **OK** button.

6. Press the **Cancel** button to close the blank **New Key Properties** dialog.

   The **Field / Key Definition** dialog for the Detail file now looks like this:

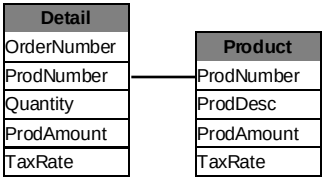


8. Press the **Close** button to close the **Field / Key Definition** dialog.
9. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

DEFINING A KEY FOR THE PRODUCT FILE

Only one field in the Product file relates to another file in the database: the ProdNumber field.

- The ProdNumber field relates to the Detail file.



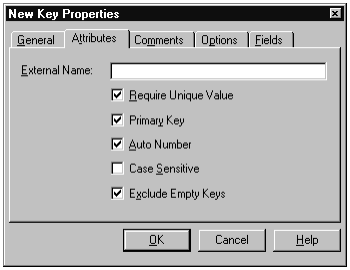
There should be no duplicate or null order numbers in the Products file; this is a *primary* key.

For each ProdNumber in the record there can be many Detail records. This is a *One to Many* relationship with the Products file a "Parent" to the Detail file.

Create the Primary Key

❑ Name the Key

1. Highlight the *Products* file in the **Files** list.
2. Press the **Field/Keys...** button.
3. Select the **Keys** tab.
4. Press the **Insert** button.
5. Type *KeyProdNumber* in the **Name** field.
6. Select the **Attributes** tab.
7. Check the **Require Unique Value** box, then check the **Primary Key** box.
8. Check the **Auto Number** box.

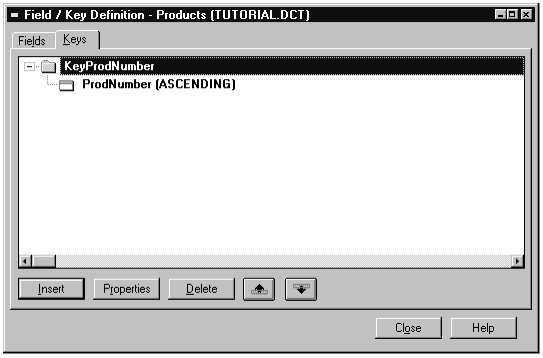


9. Select the **Fields** tab.
10. Press the **Insert** button.
11. Select *ProdNumber* then press the **Select** button.
12. Press the **OK** button.

    A blank **Key Properties** dialog appears, ready for you to specify another key.

13. Press the **Cancel** button to close the blank **Key Properties** dialog.

    The **Field / Key Definition** dialog for the Product file now looks like this:



14. Press the **Close** button to close the **Field / Key Definition** dialog.
15. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

ADDING RELATIONS

Contents

Now that all the keys are defined, we can add the relations. Once you have defined relationships, you can add Validity Checks for the fields that should only contain values that exist in another file.

These are the last steps to completing the data dictionary.

DEFINING RELATIONS FOR THE ORDERS FILE

- KeyOrderNumber relates the Orders file to the Detail file in a *One to Many* relationship.
- KeyCustNumber relates the Orders file to the Customers file in a *Many to One* relationship.

❑ Define the first relationship.

1. Highlight the *Orders* file in the **Files** list.
2. Press the **Add Relation** button.

   The default relationship **Type** is *1:MANY*, which you should accept.
3. Choose *KeyOrderNumber* from the **Primary Key** dropdown list.
4. Choose *Detail* from the **Related File** dropdown list.
5. Choose *KeyOrderNumber* from the **Foreign Key** dropdown list.
6. Press the **Map by Name** button.

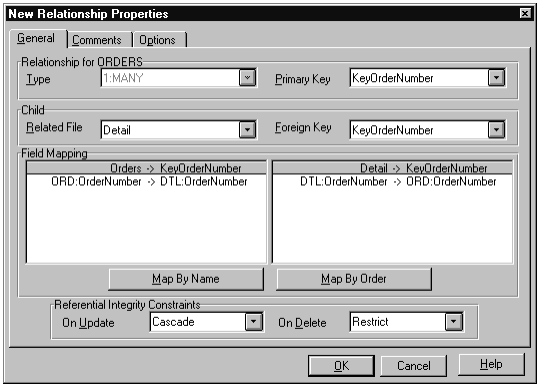
   This establishes the relationship by linking all the fields in the two keys that have the same name.

❑ Set up the Referential Integrity constraints.

1. Choose *Cascade* from the **On Update** dropdown list.

   This tells the templates to generate code to automatically update all related “Child” records when the “Parent” key field value changes.
2. Choose *Restrict* from the **On Delete** dropdown list.

This does not allow the user to delete a “Parent” record that has related “Child” records.



3. Press the **OK** button.

❑ Define the second relationship.

1. Highlight the *Orders* file in the **Files** list.
2. Press the **Add Relation** button.
3. Choose *MANY:1* from the **Type** dropdown list.

   Notice that the prompts for **Primary Key** and **Foreign Key** switch places. This happens because we are defining this relationship from the “Child” file’s viewpoint.
4. Choose *KeyCustNumber* from the **Foreign Key** dropdown list.
5. Choose *Customer* from the **Related File** dropdown list.

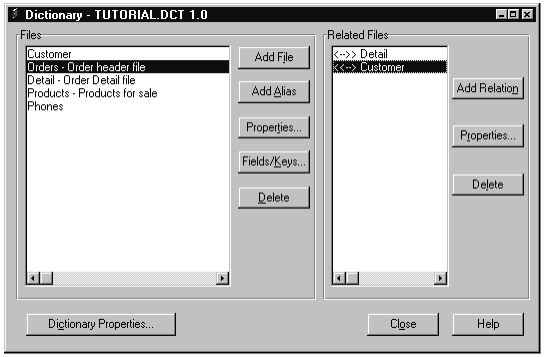
   This establishes the Customer file as the “Parent” in this relationship.
6. Choose *KeyCustNumber* from the **Primary Key** dropdown list.
7. Press the **Map by Name** button.

❑ Set up the Referential Integrity constraints.

1. Choose *Cascade* from the **On Update** dropdown list.

   Although we are deining this relationship from the “Child” file’s viewpoint, the Referential Integrity constraints are still set on the “parent” file actions.
2. Choose *Restrict* from the **On Delete** dropdown list.
3. Press the **OK** button.

   Your **Dictionary** dialog should now look like this:



4. Choose **File ➤ Save**, or press the *Save button* on the tool bar.

DEFINING RELATIONS FOR THE DETAIL FILE

Each time you define a relationship in the Dictionary Editor, you define it for both files at the same time. Therefore, since you have defined all the relationships for the Orders file, there is only one relationship left to define in this data dictionary.

The last relationship is for the Detail file.

- KeyOrderNumber relates the Orders file to the Detail file in a *One to Many* relationship. You have already defined this.
- KeyProdNumber relates the Detail file to the Product file in a *Many to One* relationship.

❑ Define the relationship.

1. Highlight the *Detail* file in the **Files** list.
2. Press the **Add Relation** button.
3. Choose *MANY:1* from the **Type** dropdown list.
4. Choose *KeyProdNumber* from the **Foreign Key** dropdown list.
5. Choose *Products* from the **Related File** dropdown list.
6. Choose *KeyProdNumber* from the **Primary Key** dropdown list.
7. Press the **Map by Name** button.

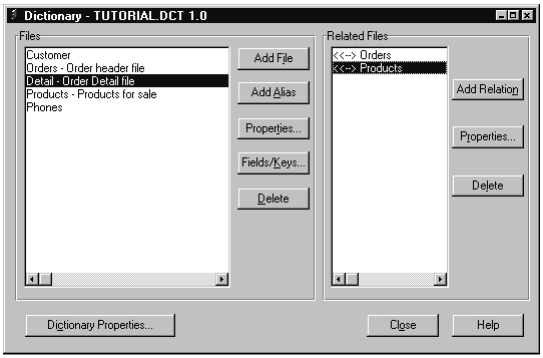
❑ Set up the Referential Integrity constraints.

1. Choose *Restrict* from the **On Update** dropdown list.

   We won't allow any changes to the product numbers.

2. Choose *Restrict* from the **On Delete** dropdown list.
3. Press the **OK** button.

   Your **Dictionary** dialog should now look like this:



4. Choose **File ➤ Save**, or press the *Save* button on the tool bar.

DEFINING RELATION-DEPENDENT VALIDITY CHECKS

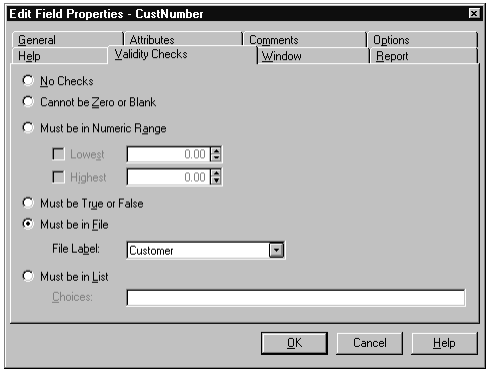
Now that all the file relationships have been defined, we can set the Validity Checks for two fields that we expect to put on update forms.

- When entering a new Order file record, we can specify that the CustNumber must match an existing record in the Customer file.
- When entering a new Detail file record, we can specify that the ProdNumber must match an existing record in the Products file.

Define the Validity Check for New Order Records

1. Highlight the *Order* file in the **Files** list.
2. Press the **Field/Keys...** button.
3. Highlight *CustNumber* and press the **Properties** button.
4. Select the **Validity Checks** tab
5. Select the **Must Be In File** radio button.
6. Choose *Customer* from the **File Label** dropdown list.

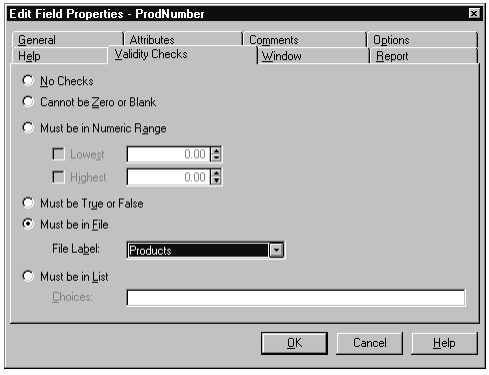
   This requires that the field can only contain values verified by getting a matching record from the Customer file. This is validated using the file relationship information, which is why this Validity Check cannot be set until the relationships have been defined.



7. Press the **OK** button.
8. Press the **Close** button to close the **Field / Key Definition** dialog.

Define the Validity Check for New Detail Records

1. Highlight the *Detail* file in the **Files** list.
2. Press the **Field/Keys...** button.
3. Highlight *ProdNumber* and press the **Properties** button.
4. Select the **Validity Checks** tab
5. Select the **Must Be In File** radio button.
6. Choose *Products* from the **File Label** dropdown list.

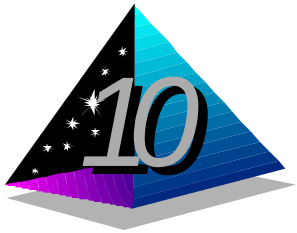


7. Press the **OK** button.
8. Press the **Close** button to close the **Field / Key Definition** dialog.
8. Press the **Close** button to close the **Dictionary Editor**.

The data dictionary is now complete. In the next chapter, we will begin building an application “from scratch” using the Application Generator.

CREATING THE APPLICATION

Contents



With the Data Dictionary complete, you now can use the Application Generator to create your application.

This chapter:

- Shows you how to create the .APP file, which stores all your work for the project.
- How to define the first (Main) procedure using the *Frame* Procedure template to create an MDI application frame.
- How to add commands to the application frame's menu bar.

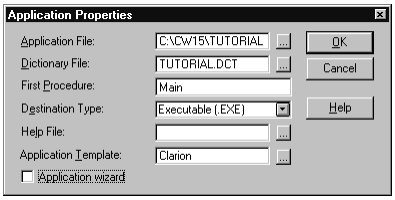
CREATING THE .APP FILE

We'll start by telling the Application Generator the name of your application and what data dictionary it uses.

1. Choose **File ➤ New** from the menu.
2. Select the **Application** tab, uncheck the **Use Quick Start Wizard** box, then press the **Create** button.

   The **Application Properties** dialog appears.
3. Press the ellipsis (...) button to the right of the **Application File** Field.
4. Select the *\CW15\TUTORIAL* directory in the **Open File** dialog, then type *TUTORIAL.APP* in the **File Name** Field.
5. Press the **OK** button to close the **Open File** dialog.
6. Type *TUTORIAL.DCT* in the **Dictionary** field.
7. Uncheck the **Application Wizard** box.

   This tutorial will not use any of the Wizards so that it can demonstrate how to use all the other tools that Clarion for Windows provides.



8. Press **OK** to close the **Application Properties** dialog.

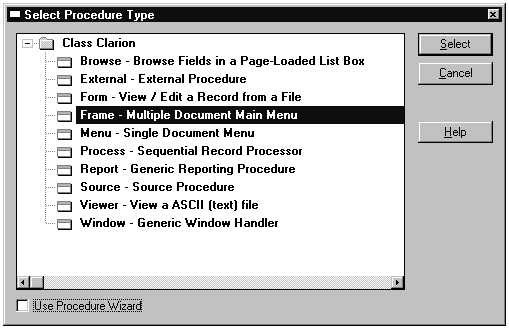
CREATING THE MAIN PROCEDURE

The **Application Tree** dialog appears. This lists all the procedures for your application in a logical procedure call tree which provides a visual guide to show the order by which one procedure calls another. You previously saw it in the Quick Start tutorial.

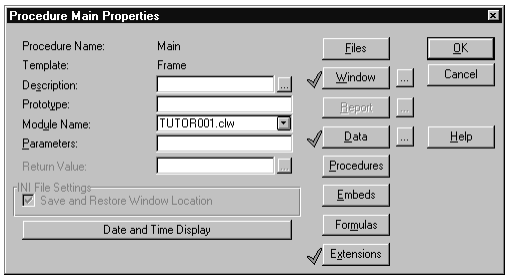
The *Main* procedure is the starting point. The tutorial application will be an MDI (Multiple Document Interface) procedure. Therefore the natural starting point is to define the *Main* procedure using the **Frame** Procedure template to create an application frame.

❑ Select the procedure type for Main:

1. With *Main* highlighted in the **Application Tree** dialog, press the **Properties** button.
2. Highlight *Frame* in the **Select Procedure Type** dialog, uncheck the **Use Procedure Wizard** box, then press the **Select** button.



The **Procedure Properties** dialog appears. It defines the functionality and data structures for the procedure.



Usually the first task when creating a procedure is to edit the main window. You can place controls, or if the procedure template has controls already, you can customize them.

The application frame *never* has controls. Windows doesn't allow it. We will, however, customize the window caption (the text that appears on its title bar). Then we will add items to the predefined menu, which is also built into the *Frame* Procedure template.

❏ Edit the Main window.

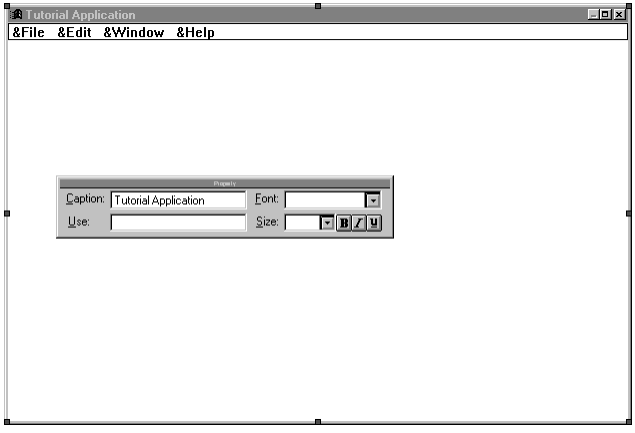
1. Press the **Window** button.

   The Window Formatter appears. Here you can visually edit the window and its controls.

2. Choose **Options ➤ Show Propertybox** (if it is not already present).

3. CLICK on the sample window's title bar to make sure it has focus.

4. Type *Tutorial Application* in the **Caption** field in the floating **Propertybox** toolbox, then press TAB.



This updates the caption bar text in the sample window. Be sure that handles appear inside the sample application frame window when you execute this step.

EDITING THE MENU

From the Window Formatter menu, you can call the Menu Editor, which allows you to add or edit menu items for the application frame window. As you add menu items, you can select the **Actions** tab to name the procedure which is called when the user chooses the menu item.

For each new procedure you name for the menu to call, the Application Generator automatically adds a "ToDo" procedure to the Application Tree. You can then define that procedure's functionality, just as you are now defining the frame's functionality.

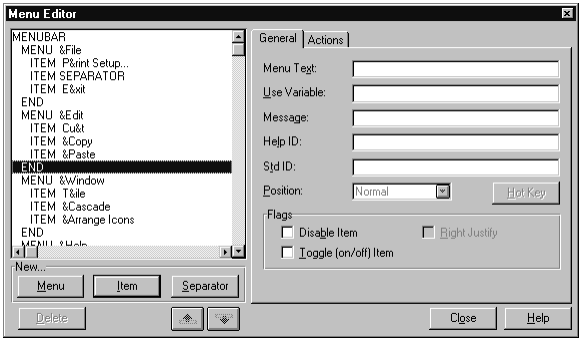
When the Application Generator generates the source code for your application, it automatically starts a new execution thread for each procedure you call from the main menu (this is required for an MDI application).

❑ Add a menu

1. From the Window Formatter menu, choose **Menu ➤ Menu Editor**.

   The Menu Editor appears. It displays the menu in hierarchical form in a list box at the left. The fields at the right allow you to name and customize the dropdown menus and menu items.

   This template already provides you with a "standard" menu. It contains basic window commands such as an **Exit** command on a **File** menu, the standard editing **Cut**, **Copy**, and **Paste** commands, and the standard window management commands commonly found in an MDI application.



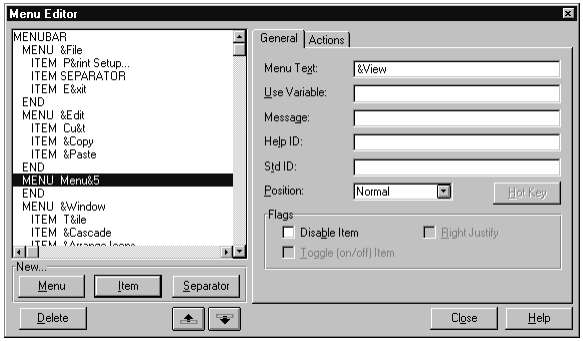
2. Highlight the second END statement (see the illustration above).

   The Menu Editor inserts new items immediately *below* the currently highlighted selection. The menu you'll add will be called **View**. It will contain three items: **Products**, **Customers**, and **Orders**. It will appear on the menu bar just before the **Window** menu.

3. Press the **Menu** button.

This inserts a new MENU statement, and its corresponding END statement.

4. Type *&View* in the **Menu Text** field.



This defines the text that appears on the menu to the end user. The ampersand (&) indicates that the following character (V) should be underlined, providing keyboard access.

❑ Add the first menu item.

1. Press the **Item** button.

   This updates the list on the left side of the dialog, changing the text of the menu you just added to "&View." It adds a new menu item—a command on the dropdown menu—under &View, and before the END statement that goes with the &View menu.

2. Type *&Customers* in the **Text** field then press TAB.

   ?ViewCustomers appears in the **Use** field. This provides an equate for the menu item so code statements can reference it. The leading question mark character (?) indicates it is a field equate label.

3. Select the **Actions** tab.

   The prompts allow you to name a procedure to execute when the end user executes the **View ➤ Customers** command.

4. Choose **Call a Procedure** from the **When Pressed** dropdown box.

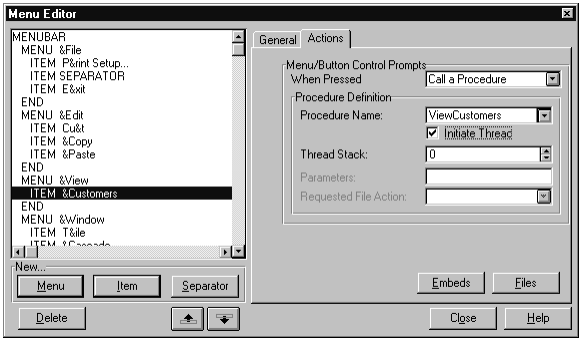
   New prompts appear to allow you to name the procedure to call and choose options.

5. Type *ViewCustomers* in the **Procedure Name** field.

   This names the procedure.

6. Check the **Initiate Thread** box.

   The *ViewCustomers* procedure will display an MDI "child" window, and you must always start a new execution thread for any MDI window called directly from the application frame. The **Thread Stack** field will default to the minimum valid value.



❑ Add the second menu item.

1. Press the **Item** button.

   This updates the list on the left side of the dialog, changing the text of the item you just added to "&Customers."

2. Type *&Products* in the **Text** field and press TAB.

   ?ViewProducts appears in the **Use** field.

   Normally, the next step is to define the action for the menu item—what happens when the end user executes it from the menu.

   You'll skip over this step for now, for this menu item only. Later, you'll create a procedure by copying it, then attaching it to this menu. Therefore, you don't want to already have a "To Do" menu of the same name.

❑ Add the third menu item.

1. Press the **Item** button.
2. Type *&Orders* in the **Text** field and press TAB.

   ?ViewOrders appears in the **Use** field.

3. Select the **Actions** tab.
4. Choose **Call a Procedure** from the **When Pressed** dropdown box.
5. Type *ViewOrders* in the **Procedure Name** field.
6. Check the **Initiate Thread** box.

❑ Close the Menu Editor and Window Formatter, then save your work.

1. Press the **Close** button to close the Menu Editor.

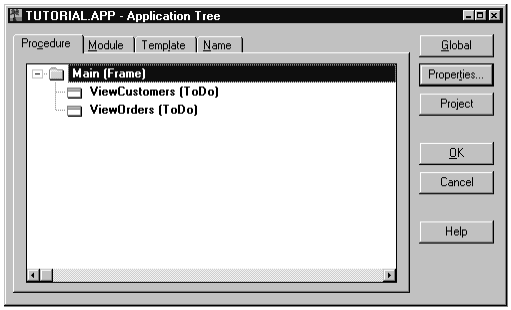
   This returns you to the Window Formatter.

2. Choose the **Exit!** menu selection and answer *Yes* when asked to save your window changes.

This returns you to the **Procedure Properties** dialog.

3. Press the **OK** button to close the **Procedure Properties** dialog.

   This returns you to the **Application Tree** dialog. It adds two new procedures marked as "(ToDo)": ViewCustomers and ViewOrders. These were the procedures you named in the Prompts dialogs.



4. Choose **File ➤ Save**, or press the *Save* button on the toolbar.

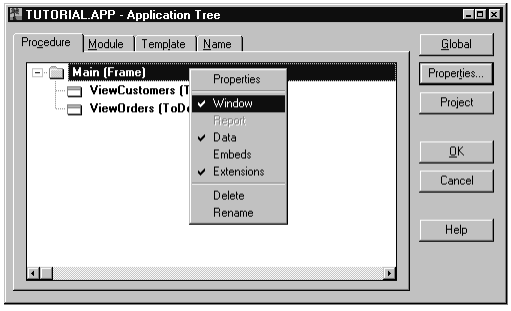
ADDING AN APPLICATION TOOLBAR

You can easily add toolbar buttons to a window. To demonstrate, you'll place a toolbar containing three buttons, each calling the same procedures you just named.

For simplicity, these will be plain text buttons. Please see the *User's Guide* for instructions on adding icon buttons.

❑ Call the Window Formatter and create the tool bar.

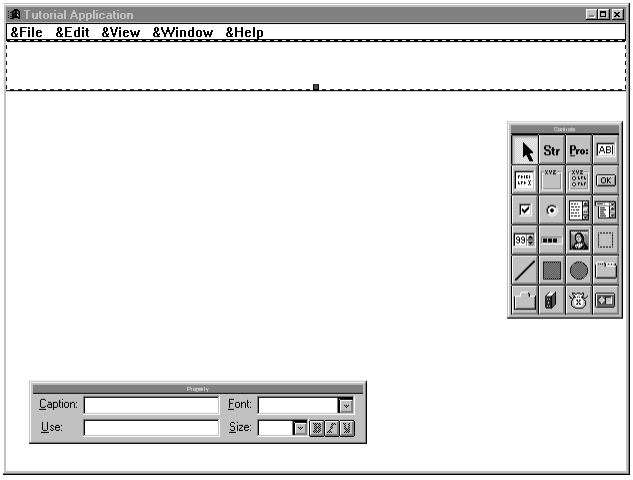
1. Highlight the *Main* procedure.
2. RIGHT-CLICK to display the popup menu.



3. Choose the **Window** menu item.

4. In the Window Formatter's menu, choose **Toolbar ➤ New Toolbar**.

   This adds an area for the toolbar—outlined by a dotted line—to the sample window. At the same time, the **Controls** toolbox appears. You can add any type of control to your toolbar by CLICKING on a tool icon in the Controls toolbox, then CLICKING in your toolbar.



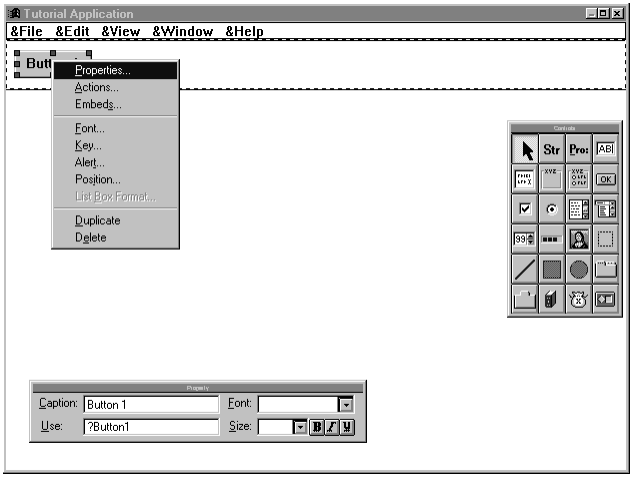
❑ Place the first button.

1. CLICK on the button tool.

   The button tool is the one that looks like an "OK" button.

2. CLICK in the sample window toolbar area, just below the upper left corner.

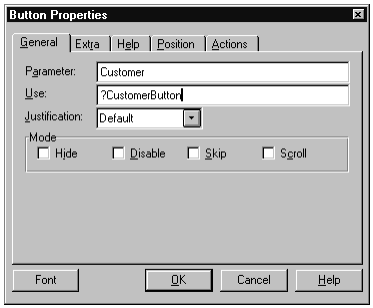
3. RIGHT-CLICK the button you just placed, then choose **Properties** from the popup menu.



The **Button Properties** dialog appears.

3. Type *Customers* in the **Parameter** field.
4. Type *?CustomerButton* in the **Use** field.

   This provides the field equate label for the button. We included the word "button" for code readability.



5. Select the **Actions** tab.
6. Choose **Call a Procedure** from the **When Pressed** dropdown box.
7. Choose **ViewCustomers** from the **Procedure Name** dropdown box.

   This is the procedure name you typed for the **View ➤ Customers** menu item. Pressing the button will call the same procedure. Often, a command button on a toolbar serves as a quick way to execute a menu command.

9. Check the **Initiate Thread** box.
10. Press the **OK** button.

❏ Place the second button.

1. CLICK on the button tool.
2. CLICK in the sample window toolbar area, next to the first button.

   A button appears, labelled Button2.

3. RIGHT-CLICK the button you just placed, then choose **Properties** from the popup menu.
4. Type *Products* in the **Button Text** field.
5. Type *?ProductsButton* in the **Use** field.
6. Press the **OK** button to close the **Button Properties** dialog.

   Normally you attach an action to the button at this point. Skip this step for now, for this button only. Later, we'll copy a procedure, then call it at the point in the generated source code which handles what to do when the end user presses the button.

❑ Place the third button.

1. CLICK on the button tool.
2. CLICK in the sample window toolbar area, next to the second button.
3. RIGHT-CLICK the button you just placed, then choose **Properties** from the popup menu.
4. Type *Orders* in the **Button Text** field.
5. Type *?OrdersButton* in the **Use** field.
6. Select the **Actions** tab.
7. Choose **Call a Procedure** from the **When Pressed** dropdown list.
8. Choose *ViewOrders* from the **Procedure Name** dropdown list.

   This is the procedure name you typed for the **View ➤ Orders** menu item.
9. Check the **Initiate Thread** box.
10. Press the **OK** button.

❑ Resize and align the buttons.

The Window Formatter has a set of alignment tools that easily allow you to line up and resize your window controls.

1. With the *Orders* button still selected, CTRL+CLICK on the *Products* button.

   This gives both buttons "handles" and the *Products* button has the Red handles that indicate it has focus.

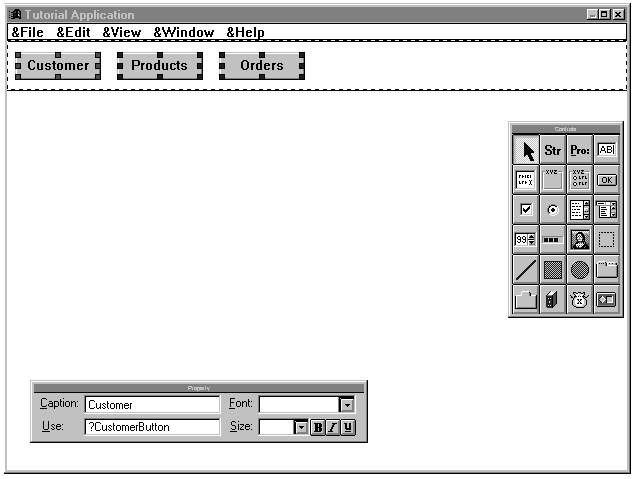
   CTRL+CLICK is the "multi-select" keystroke that allows you to perform actions on several controls at once. Once multiple controls are selected, you can move them all by DRAGGING on any one of the selected controls, or you can use any of the **Alignment** menu's tools on the entire group.
2. With both buttons still selected, CTRL+CLICK on the *Customers* button.

   Now all three buttons have "handles" and the *Customers* button has the Red handles that indicate it has focus.
3. Choose **Alignment ➤ Make Same Size**.

   This will make all three buttons the same size as the one with the Red handles (the *Customers* button).
4. Choose **Alignment ➤ Align Top**.

   This aligns all three buttons with top of the *Customers* button.
5. Choose **Alignment ➤ Spread Horizontally**.

This spaces all three buttons apart equally.



❑ Close the Window Formatter and save your work

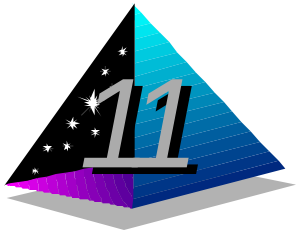
1. Choose the **Exit!** menu selection and answer *Yes* when asked to save your window changes.

   This returns you directly to the **Application Tree** dialog. It still contains the same two procedures marked as "(ToDo)": *ViewCustomers* and *ViewOrders*.

2. Choose **File ➤ Save**, or press the *Save* button on the toolbar.

CREATING A BROWSE

Contents



In this chapter, you'll create a browse window similar to the one created for you by the Quick Start Wizard. The Application Generator uses the same templates, which generate the same basic code—but doing it this way, you'll have a chance to "do it from scratch."

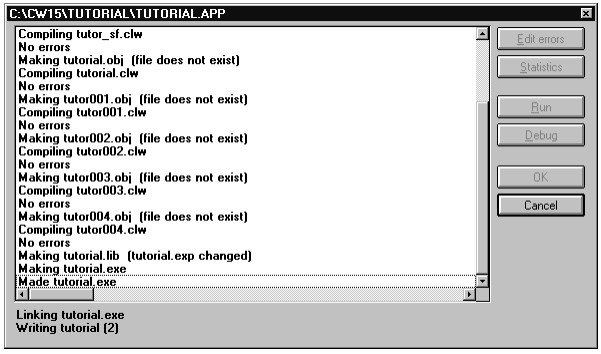
You'll start with a Customer browse window—but first, try compiling the application you created in the last chapter.

TESTING AN APP UNDER DEVELOPMENT

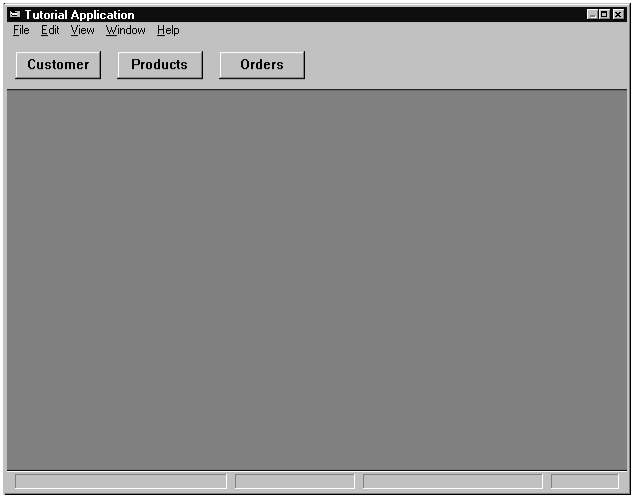
1. With the **Application Tree** dialog open, choose **Project ➤ Run**, or press the *Run* button on the tool bar.

   The Application Generator generates the source code, displaying its progress in a message window, procedure by procedure.

   Then the Make window appears, showing you the progress of the build as the compiler and linker do their work.



   Then your Application window appears. It should look something like this:



2. Choose one of the buttons on the toolbar, or one of the items on the **View** menu.

   The following message appears:

   This capability allows you to incrementally test your application, whether you have designed all the procedures or not.

   You'll fill in their functionality, starting in this chapter.

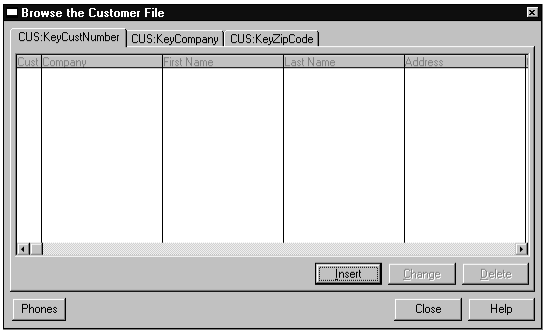
3. Press the **OK** button to close the message box.

4. Choose **File ➤ Exit** to close the Tutorial application.

   Throughout the rest of this tutorial, feel free to *Make and Run* the developing application at any point that the tutorial instructs you to save the file.

CREATING THE CUSTOMER BROWSE WINDOW

Back in the Quick Start Tutorial chapter, you will recall that the Quick Start Wizard created a window for the *Customer* file Browse procedure, that looked like this:



Now you'll create a similar one using the *Browse* Procedure template:

❑ Select the procedure type for the ViewCustomers procedure:

1. Highlight *ViewCustomers* in the **Application Tree** then press the **Properties** button.
2. Select the *Browse* Procedure template in the **Select Procedure Type** dialog, uncheck the **Use Procedure Wizard** box, then press the **Select** button.

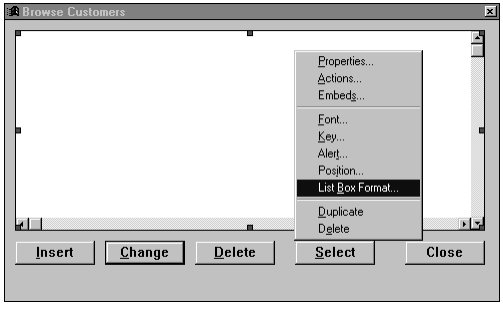
   The **Procedure Properties** dialog appears.

❑ Edit the Browse procedure.

1. In the **Procedure Properties** dialog, press the **Window** button.
2. CLICK in the window's title bar to give it focus.
3. Type *Browse Customers* in the **Caption** field in the floating **Propertybox** toolbox, then press TAB.

❑ Prepare to format the list box.

1. RIGHT-CLICK the list box in the window, then choose **List Box Format...** from the popup menu.

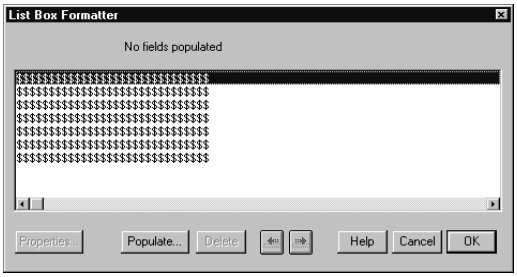


This opens the List Box Formatter.

Populating and Formatting a List Box Control

Using the List Box Formatter, you can *populate* and *format* the data dictionary fields that appear in the columns of the list.

❑ Using the **Select Field** dialog, you select a field from the data dictionary, which the List Box Formatter then defines as a column in the list box.

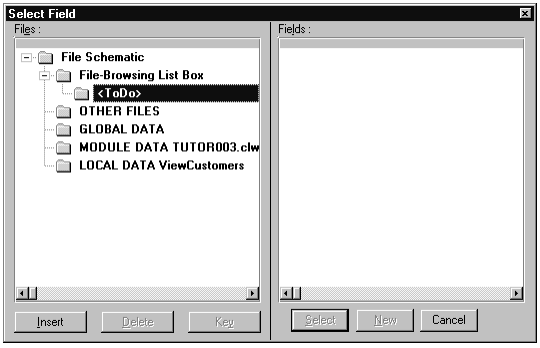


1. Press the **Populate** button.

   The **Select Field** dialog appears. This provides access to the files defined in the data dictionary. The **Files** list displays all the files selected for use in this proceure in a hierarchical arrangement (the *File Schematic*), which includes the browse list box control.

❑ Select the file and fields to place in the browse list box control.

1. Highlight the "ToDo" item below the **File-Browsing List Box** and press the **Insert** button.

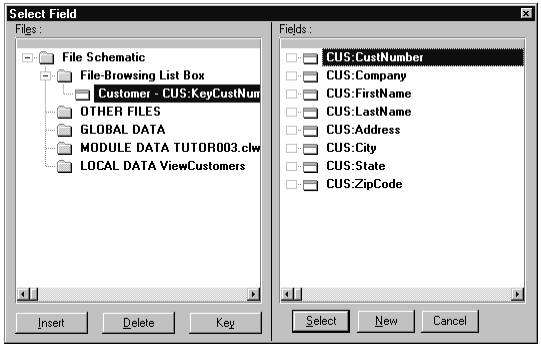


2. Highlight the *Customer* file in the **Insert File** dialog, then press the **Select** button.

   This adds the file to the **File Schematic** dialog. The **Select Field** dialog now lists the file and its fields.

3. Press the **Key** button.

4. Highlight *KeyCustNumber* in the **Change Access Key** dialog and press the **Select** button.



5. Highlight *CUS:Company* in the **Fields** list, then press the **Select** button.

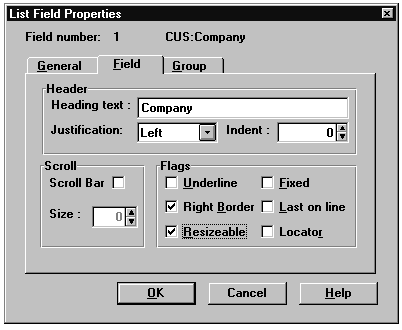
   This returns you to the List Box Formatter with the selected field added to the list.

❑ Apply special formatting to the first field:

1. Press the **Properties** button.

   The **List Field Properties** dialog appears, allowing you to format the appearance of selected field.

2. Select the **Field** tab then check the **Right Border** and **Resizeable** boxes.



3. Press the **OK** button.

   This adds a resizable right vertical border to the field at runtime.

❑ Populate the second field.

1. Press the **Populate** button.
2. Highlight *CUS:FirstName* in the **Fields** list, and press the **Select** button.

3. Press the **Properties** button.
4. Select the **Field** tab then *uncheck* the **Right Border** and **Resizeable** boxes.

   The List Box Formatter automatically "carries forward" these formatting options from the last field you added, making it very simple to add multiple fields with similar formatting options. In this case, this deletes the column divider between this and the next column, which will be the *LastName* field.
5. Press the **OK** button.

❏ Populate the third field.

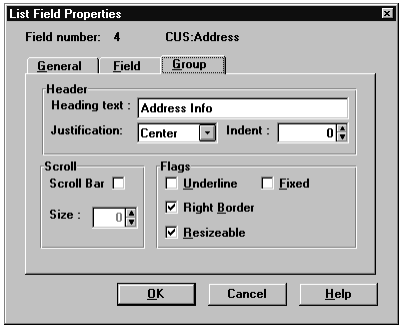
1. Press the **Populate** button.
2. Highlight *CUS:LastName* in the **Fields** list, then press the **Select** button.
3. Press the **Properties** button.
4. Select the **Field** tab then check the **Right Border** and **Resizeable** boxes.

   This once again adds the resizable column divider between this and the next column.
5. Press the **OK** button.

❏ Populate the fourth field.

1. Press the **Populate** button.
2. Highlight *CUS:Address* in the **Fields** list, then press the **Select** button.
3. Press the **Properties** button.
3. Select the **Group** tab.
4. Press the **OK** button when asked to confirm that you wish to create a new group.

   By creating a new group, in which you'll place the address information, you can add a group header. This appears *above* the field headers, and visually links the data in the columns beneath.
5. Type *Address Info* in the **Heading Text** field.



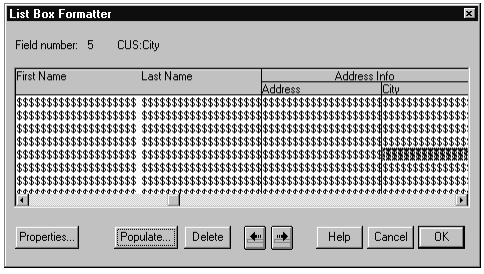
This provides the text for the group header. Any fields appearing to the right of this one will be included in the group, until you define another group.

7. Press the **OK** button to close the **List Field Properties** dialog.

❏ Populate the fifth field.

1. Press the **Populate** button.
2. Highlight *CUS:City* in the **Fields** list, and press the **Select** button.

   As you work, the List Box Formatter updates its sample window, and provides a rough estimation of how your list box will look:



❏ Populate the sixth field.

1. Press the **Populate** button.
2. Highlight *CUS:State* in the **Fields** list, and press the **Select** button.

❏ Populate the seventh field, and exit the List Box Formatter.

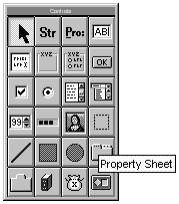
1. Press the **Populate** button.
2. Highlight *CUS:ZipCode* in the **Fields** list, and press the **Select** button.
3. Press the **OK** button to close the List Box Formatter.

Adding the Tabs

When the Quick Start Wizard created this procedure it had tab controls that changed the list's sort order depending on which tab was selected. Therefore, we'll add this functionality right now to show how easy it is to accomplish!

❑ Add the Property Sheet and the first tab.

1. CLICK on your window's title bar to place the red "handles" on your window design.
2. Place the mouse cursor diectly over the middle handle on the top then DRAG it up to create some room at the top.
3. CLICK on the *Property sheet* control in the floating **Controls** toolbox (it's the one that looks like several file folders and is right next to the circle).



4. CLICK above and to the left of the List box to place the property sheet and one Tab control.
5. DRAG the "red handle" at the bottom *left*-hand corner so that it appears just below and to the left of the Insert button.
6. DRAG the "red handle" at the bottom *right*-hand corner so that it appears just below and to the right of the Close button.

   This resizes the property sheet so that it appears as though the list box and buttons are on the tab. In fact, they are not, and we don't want them to be, since we want all these controls to be visible no matter which tab the user selects.
7. CLICK on the "Tab 1" tab.
8. Type *KeyCustNumber* in the **Caption** field in the floating **Propertybox** toolbox, then press TAB.

   This changes the tab's text. This tab text can be anything, but naming the key also names the sort order it will display.

❑ Add the rest of the tabs.

1. CLICK on the *Tab control* in the floating **Controls** toolbox (it's the one that looks like a single file folder).

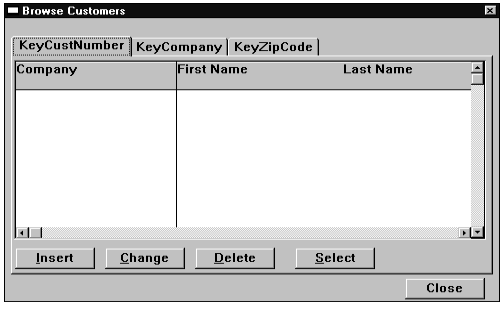
2. CLICK immediately to the right of the *KeyCustNumber* tab to place the next Tab control.
3. Type *KeyCompany* in the **Caption** field in the floating **Propertybox** toolbox, then press TAB.
4. CLICK on the *Tab control* in the floating **Controls** toolbox (it's the one that looks like a single file folder).
5. CLICK immediately to the right of the *KeyCompany* tab to place the next Tab control.
6. Type *KeyZipCode* in the **Caption** field in the floating **Propertybox** toolbox, then press TAB.
7. CLICK on the Close button then SHIFT+CLICK and DRAG the button straight down below the tab.

   DRAGGING a control with the SHIFT key depressed allows you to move the control in only one direction; if you start moving it down it will only move up and down, but if you start moving it to either side it will only move side to side.

Testing the Customer Browse

The Window Formatter provides a test mode which allows you to preview just how your window looks on the desktop. Since this is an MDI window, it appears inside the Window Formatter frame.

1. Choose **Preview!** on the Window Formatter's menu bar.



2. Press the **Close** button inside the test window to return to the Window Formatter.

Setting the Sort Orders

Now that the tabs are there, we need to tell the list box what alternate sort orders to use and when.

1. RIGHT-CLICK on the list box then choose **Actions** from the popup menu.

   The list box is actually a *BrowseBox* Control template that has been placed in the *Browse* Procedure template's default window design in the Template Registry (see the *User's Guide* for more information on the Template Registry). This means that it has associated prompts which tell it how to populate the list and what actions to perform.

   The prompts that appear on the **Actions** tab come directly from the templates (in this case, the *BrowseBox* Control template). This is how you communicate to the templates exactly what code they need to generate to give you the behavior you ask for (and nothing else). These prompts, their meanings and uses, are all covered in the *User's Guide* and in the on-line help for each window in which they appear.

2. Select the **Conditional Behavior** tab.

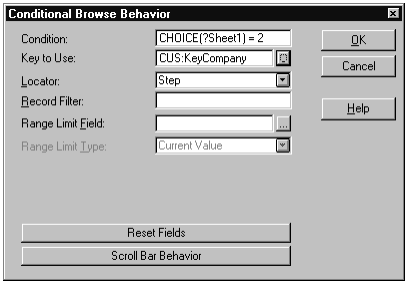
3. Press the **Insert** button.

4. Type *CHOICE(?Sheet1) = 2* in the **Condition** field.

   This sets the condition under which the alternate sort order will be used. This expression uses the Clarion language CHOICE function (see the *Language Reference*) to detect when the user has selected the second tab on the sheet. The generated code will use this expression in a conditional statement that will change the sort order at runtime.

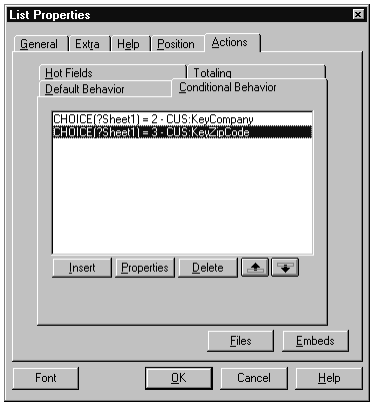
5. Press the ellipsis button (...) next to the **Key to Use** field.

6. Highlight *CUS:KeyCompany* then press the **Select** button on the **Select Key** dialog.



Now, when the user selects the second tab, the *BrowseBox* Control template will generate code to switch to the key on the *Company* field. It doen't need to know what to do for the first tab, because that always uses the Access Key we set in the File Schematic.

7. Press the **OK** button.
8. Press the **Insert** button.
9. Type *CHOICE(?Sheet1) = 3* in the **Condition** field.
10. Press the ellipsis button (...) next to the **Key to Use** field.
11. Highlight *CUS:KeyZipCode* then press the **Select** button on the **Select Key** dialog.
12. Press the **OK** button.



13. Press the **OK** button to close the **List Properties** dialog.

Closing the Customer Browse

1. Choose **Exit!** on the Window Formatter's menu bar, and save your window changes when prompted to do so.
2. Press the **OK** button in the **Procedure Properties** dialog to close it.
3. Choose **File ➤ Save**, or press the *Save* button on the toolbar to save your work.

THE BROWSE UPDATE PROCEDURE

Contents

In the last chapter, we formatted the Customer Browse procedure's list box and added tab controls to change the sort order. To finish the basic procedure, we name the Update procedure. This is the procedure that handles the action for the Insert, Change, and Delete buttons.

NAMING THE UPDATE PROCEDURE

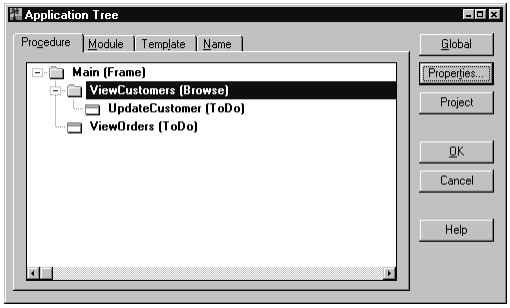
❑ Add a "ToDo" procedure.

1. Highlight *ViewCustomers* in the **Application Tree** dialog, then press the **Properties** button.
2. Type *UpdateCustomer* in the **Update Procedure** entry box at the bottom of the **Procedure Properties** dialog.

   This names a new procedure, which then appears in the **Application Tree**.
3. Check the **Allow Edit via Popup** box.

   This enables the user to RIGHT-CLICK on the list box and choose Insert, Change, or Delete from a popup menu in addition to using the command buttons.
4. Press the **OK** button to close the **Procedure Properties** dialog.

   The new procedure appears below the ViewCustomers procedure.



Notice that you didn't have to start a new execution thread for the procedure. You want it to run on the same thread as the browse, so that the end user can't open a form window to change a record, then activate the browse window again, and open another form on the same record. In other words, you don't want an end user trying to change the same record twice at the same time!

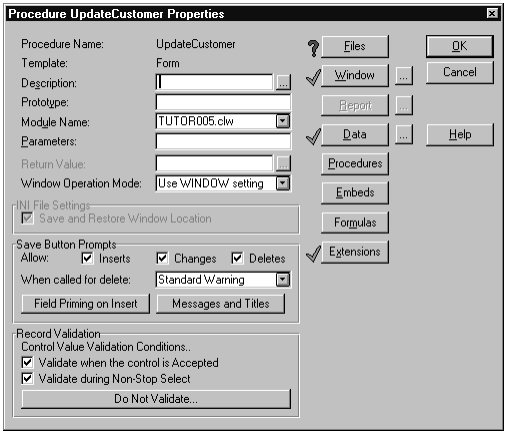
CREATING THE UPDATE PROCEDURE

The Update Procedure should use the Form Procedure template to create a procedure that the end user can use to maintain a record. It should provide a prompt and entry control for each field in the record.

❑ Select the procedure type for UpdateCustomer.

1. Highlight *UpdateCustomer* in the **Application Tree** dialog, then press the **Properties** button.
2. Highlight the *Form* Procedure template, uncheck the **Use Procedure Wizard** box, then press the **Select** button.

   The **Procedure Properties** Window appears. You'll accept the defaults for the Form procedure template. The *User's Guide* and on-line help describes the customization options available for each **Procedure Properties** dialog (they vary for each Procedure template).

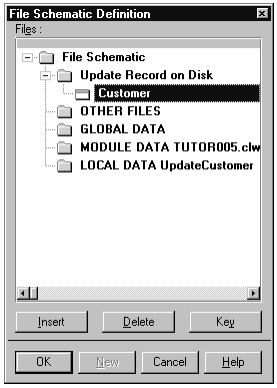


3. Press the **Files** button to name the file the Form will update.

   The **File Schematic Definition** dialog appears.

4. Highlight the "ToDo" item below the **Update Record on Disk** and press the **Insert** button.

5. Highlight the *Customer* file in the **Insert File** dialog, then press the **Select** button.



6. Press the **OK** button to return to the **Procedure Properties** window.
7. Press the **Window** button to design your form.

   The default window design contains three fields for you already. The *OK* button will close the dialog, accepting the end user's input and writes the Customer file record to disk. The *Cancel* button closes the form without updating. The string field provides an action message to inform the end user what action they are taking on the record.

Populating the fields.

Placing the fields in a window is called *populating* it.

1. Choose **Options ➤ Show Fieldbox**.

   This displays a floating toolbox containing all the fields from all the files specified in your procedure's File Schematic. These fields are all ready to populate onto your window design.

2. CLICK on *CustNumber* in the floating **Fieldbox** then move the cursor over the window design.

   The cursor changes to a crosshair and a "little book" which indicates the field comes from the data dictionary.

3. CLICK in the upper left corner of your window design.

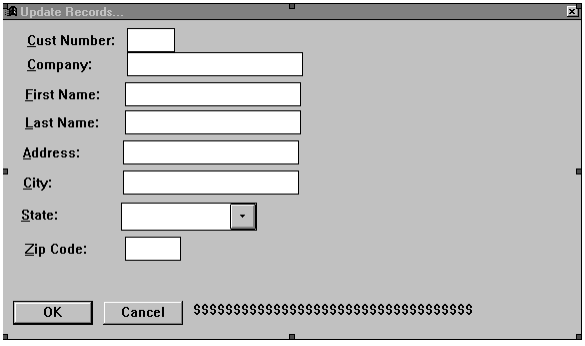
This places both the field and its prompt. These default to whatever you specified in the data dictionary.

4. CLICK on *Company* in the floating **Fieldbox**.
5. CLICK just below the prompt placed for the first field.
6. CLICK on *FirstName* in the floating **Fieldbox**.
7. CLICK just below the prompt placed for the last field.

   This places both the field and its prompt.
8. CLICK on *LastName* in the floating **Fieldbox**.
9. CLICK just below the prompt placed for the first name.
10. CLICK on *Address* in the floating **Fieldbox**.
11. CLICK just below the prompt placed for the last name field.
12. CLICK on *City* in the floating **Fieldbox**.
13. CLICK just below the prompt placed for the address field.
14. CLICK on *State* in the floating **Fieldbox**.
15. CLICK just below the prompt placed for the city.

    This places the prompt and the drop list. We pre-defined this field as a LIST control with the DROP attribute in the data dictionary. Since it has a pre-defined set of valid entries, we don't have to format it.
16. CLICK on *ZipCode* in the floating **Fieldbox**.
17. CLICK just below the prompt placed for the state.

    The form window now looks something like this:



Moving and Aligning Fields

For a professional look, we need to move these fields around and align the sides and bottoms of all the fields in the screen.

❑ Move the fields to their approximate positions.

1. CLICK on the *State* drop list.

   Its handles should appear when you CLICK on it.

2. SHIFT+DRAG the *State* drop list to the left so it is closer to its prompt.

3. CTRL+CLICK on the *State* prompt.

   When you CTRL+CLICK on a control, the previously selected control's handles turn blue while the newly selected control's handles are red. You now have two controls selected.

4. Move the *State* drop list and prompt (CLICK and DRAG on either of the two selected controls) up and to the right of the *City* controls.

   Once multiple controls are selected, you can move them as a group the same way you would move one individual control.

5. CLICK on the *ZipCode* entry box.

6. SHIFT+DRAG the *ZipCode* entry box to the left so it is closer to its prompt.

7. CTRL+CLICK on the *ZipCode* prompt.

8. Move the *ZipCode* entry box and prompt up and to the right of the *City* and *State* controls.

   You may need to make the window a little larger to accomplish this.

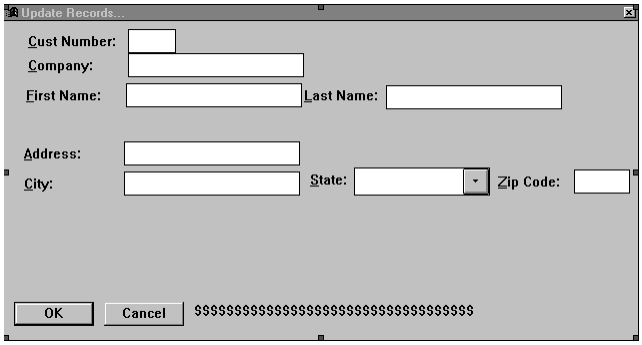
9. CLICK on the *LastName* entry box.

10. SHIFT+DRAG the *LastName* entry box to the left so it is closer to its prompt.

11. CTRL+CLICK on the *LastName* prompt.

12. Move the *LastName* entry box and prompt up and to the right of the *FirstName* controls.

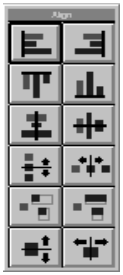
    Now your window should appear something like this:



❑ Align the fields to their final positions.

1. Choose **Options ➤ Show Alignbox**.

   The Window Formatter has a floating **Alignbox** toolbox that contains the same set of alignment tools that are also available through the **Alignment** menu.



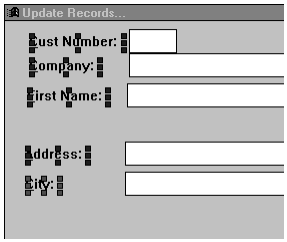
2. CLICK on the first prompt in the upper left corner.

   This should be the *CustNumber* prompt. Its handles should appear when you click it.

3. CTRL+CLICK on the four prompts immediately below the first.

   As you CTRL+CLICK on each control in turn, the previously selected controls' handles turn blue while the newest selected control's handles are red. The control with the red handles provides the "base point" for the alignment operation. All the other selected controls are aligned in relation to the control with the red handles.

   With all five prompts selected, it should look like this:



4. Press the **Align Left** button (the top left button) in the floating **Alignbox** toolbox.

   The controls all line up along their left edges, based on the position of the last item selected (the one with the red handles).

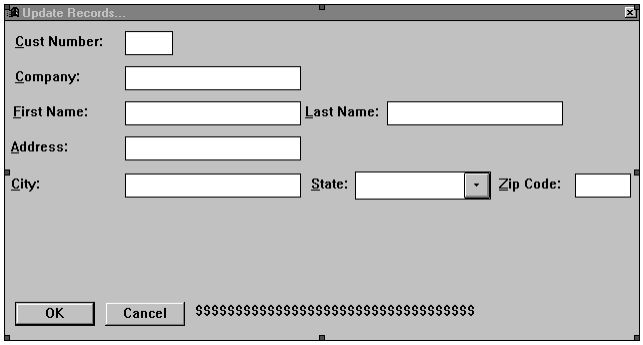
5. Press the **Spread Vertically** button in the floating **Alignbox** toolbox.

   The controls all evenly spread themselves between the top and bottom controls selected.

6. CLICK on the first entry control (this should be the *CustNumber* entry field).
7. CTRL+CLICK on the entry controls immediately below it.
8. Press the **Align Left** button in the floating **Alignbox** toolbox.
9. CLICK on the *LastName* entry box.
10. CTRL+CLICK on the three controls to its left (its prompt, the *FirstName* entry field and prompt).
11. Press the **Align Horizontally** button in the floating **Alignbox** toolbox.

    This aligns the controls in a neat row.
12. CLICK on the last control in the bottom row (this should be *ZipCode* entry box).
13. CTRL+CLICK on the five controls to its left (its prompt, and the prompts and controls for *State* and *City*).
14. Press the **Align Horizontally** button in the floating **Alignbox** toolbox.
15. In the same manner, use the **Align Horizontally** tool to align the *CustNumber*, *Company*, and *Address* entry boxes with their respective prompts.

    The window should now look something like this:



The form window is almost done. Now we will add a browse list box for the related Phones file records.

Adding a BrowseBox Control Template

Control templates generate all the source code required to create and maintain a control on your window. Here, you place a *BrowseBox* Control template that displays the records from the *Phones* file that are related to the current *Customer* record.

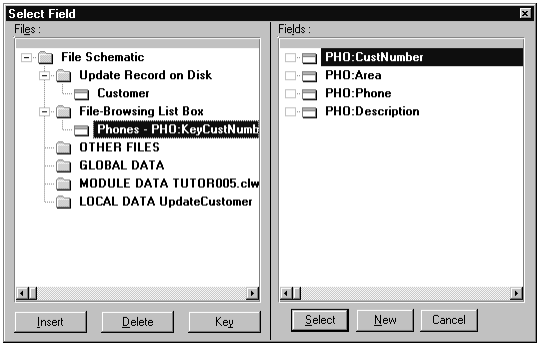
❑ Place the Control Template

1. Choose **Populate ➤ Control Template**, or CLICK on the Control Template tool in the floating **Controls** toolbox (last tool icon on the right, bottom row).
2. In the **Select Control Template** dialog, Highlight the *BrowseBox* Control template, then press the **Select** button. The cursor changes to a crosshair and "little book."
3. CLICK just below the *City* entry box to place the control.

   The List Box Formatter appears, ready for you to choose fields.

❑ Place the Phones file fields in the List Box Formatter

1. Press the **Populate** button.
2. Select the "ToDo" item below the **File-Browsing List Box** and press the **Insert** button.
3. Highlight the *Phones* file in the **Insert File** dialog, then press the **Select** button.
4. Highlight the *Phones* file in the **Files** list, then press the **Key** button.
5. Highlight *KeyCustNumber* in the **Change Access Key** dialog, then press the **Select** button.
6. Highlight *PHO:CustNumber* in the **Fields** list, then press the **Select** button.



8. Press the **Populate** button.
9. Highlight *PHO:Area* in the **Fields** list, then press the **Select** button.

11. Press the **Populate** button.

12. Highlight *PHO:Phone* in the **Fields** list, then press the **Select** button.

14. Press the **Populate** button.

15. Highlight *PHO:Description* in the **Fields** list, then press the **Select** button.

    As an optional step, resize the columns in the List Box Formatter sample window to make them wide enough for the column headers. To resize the columns, just DRAG them with the mouse.

17. Press the **OK** button to close the List Box Formatter.

    This places the formatted List box on the window at the position we specified. This may expand the window. If so, resize the List box by DRAGGING its handles, then move the OK, Cancel, and Message string controls down to the new bottom of the window.

❑ Set up the control template's record range limits.

1. RIGHT-CLICK the list box you just placed, and select **Actions** from the popup menu.

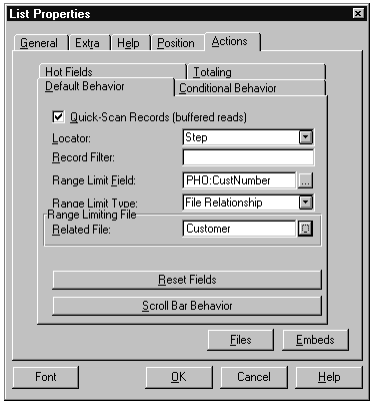
2. Press the ellipsis (...) button next to the **Range Limit Field**.

3. Highlight the *PHO:CustNumber* field in the **Key Components** list, then press the **Select** button.

4. Choose **File Relationship** from the **Range Limit Type** dropdown list.

5. Press the ellipsis (...) button next to the **Related File** field then select *Customer* from the **Select File** dialog.

   This identifies the Customer file as the related file. These steps limit the records displayed in the list box to only those records related to the currently displayed Customer file record.



6. Press **OK** to close the **List Properties** dialog.

Adding the BrowseUpdateButtons Control Template

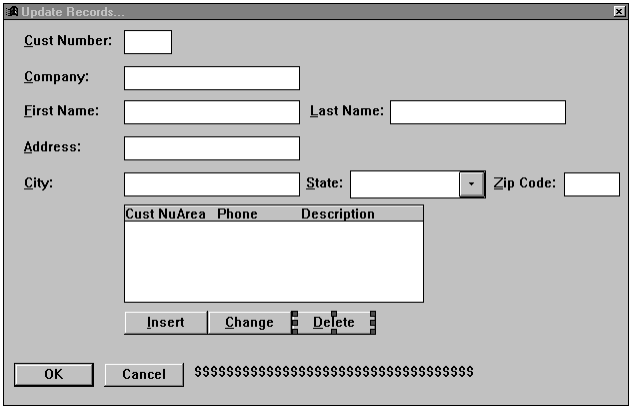
Next add the standard Insert, Change and Delete buttons for the list box control. We'll later add a form procedure for adding or editing a phone.

❏ Place another type of Control Template.

1. Choose **Populate ➤ Control Template**, or click on the Control Template tool in the **Controls** toolbox (last tool icon on the right, bottom row).
2. Highlight the *BrowseUpdateButtons* Control template, then press the **Select** button.

   The cursor changes to a crosshair and "little book."
3. CLICK below the left end of the list box.

   The **Insert**, **Change**, and **Delete** buttons appear. At this point the window should look something like this:



❏ Name the Update Procedure.

1. RIGHT-CLICK on the **Delete** button and choose **Actions** from the popup menu.
2. Type *UpdatePhones* in the **Update Procedure** box in the **Prompts** dialog.

   Naming the Update Procedure for one button names it for all three buttons in the set.
3. Press the **OK** button.
4. Choose **Exit!** to close the **Window Formatter**.

5. Press the **OK** button to close the **Procedure Properties** dialog.
6. Choose **File ➤ Save**, or press the *Save* button on the tool bar.

Stealing the Phones Update Form

You can now view the Phones list in the Customer form. However, you still need a way to update the Phones record. You could add another Form procedure for that purpose. However, there is an easier way. Since the application we created in the Quick Start Tutorial chapter also uses the Phones file, we can "steal" the procedure from there!

1. Choose **File ➤ Import from Application...**.
2. Highlight *QWKTUTOR.APP* in the **Select application to import from** dialog, then press the **OK** button.

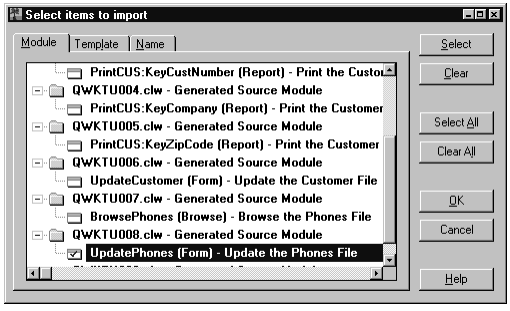
   The following warning appears. Since we know that the only procedure we want is one whose file definition was inported from this application's dictionary, we are safe in continuing with the import.

3. Press the **OK** button.

   The **Select Items to Import** dialog appears.

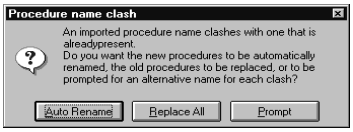
4. Highlight the *UpdatePhones* procedure then press the **Select** button.

   This places a check mark in the icon next to the procedure name indicating that it has been selected for import. You can import multiple procedures at one time, if you choose.



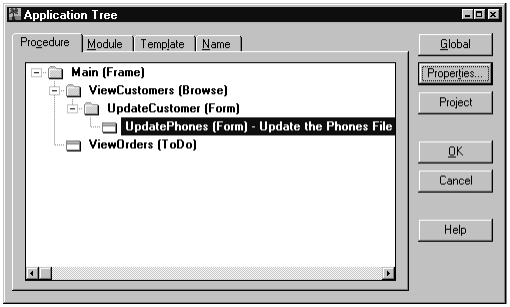
5. Press the **OK** button.

   Now another warning appears that we are importing a procedure with a name our application already is using.



6. Press the **Replace All** button.

   This implements the procedure import, giving you an application tree that looks like this:

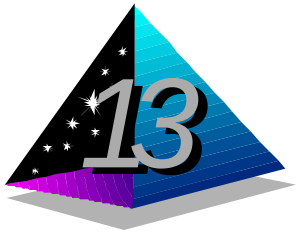


7. Choose **File ➤ Save**, or press the *Save* button on the tool bar.

   You can now make and run your application to test its functionality at this point, if you choose.

CREATING THE PRODUCT BROWSE

Contents



Now that we've created the Customer browse, we can reuse much of that work for the next procedure by copying the procedure, then changing its fields. In this chapter, you'll copy the *ViewCustomers* procedure to create the *ViewProducts* procedure.

You will also use "Embed points" to write "embedded source code" to call the *Viewproducts* procedure from your application's menu and toolbar. This will introduce you to the numerous points at which you can add a few (or many) lines of your own source code to add functionality to any procedure.

COPYING A PROCEDURE

As you recall, when you created your **View ➤ Products** menu item, and the toolbar button labelled "Products," you didn't specify a procedure to call when the end user executed them. We'll start by creating the procedure to call.

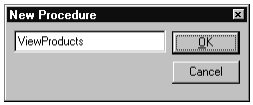
1. Highlight the *ViewCustomers* procedure in the **Application Tree** dialog.

   This is the procedure you will copy.

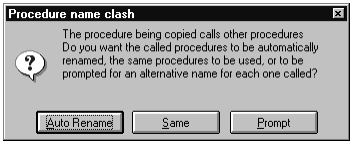
2. Choose **Procedure ➤ Copy...**.

   The **New Procedure** dialog appears.

3. Type *ViewProducts* in the entry box then press the **OK** button.



   Because the *UpdateCustomer* procedure is nested under the *ViewCustomers* procedure (the one you are copying), the **Procedure name clash** dialog appears. This offers you options on how to handle the clashing procedures.

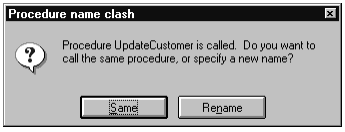


4. Press the **Prompt** button.

   By pressing the **Prompt** button, you tell the Application Generator to let you rename all the clashing procedures.

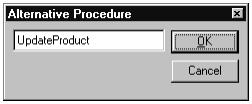
   Another warning message box appears to inform you of a specific duplicate procedure name.

5. Press the **Rename** button.

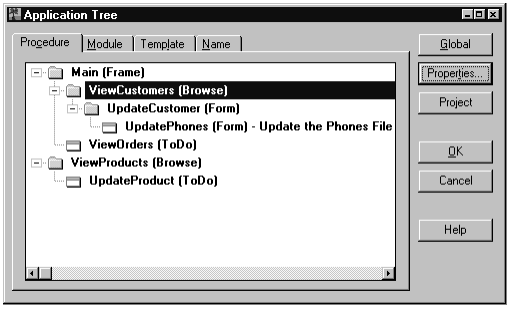


   The **Alternate Procedure** dialog appears.

6. Type *UpdateProduct* in the entry box, then press the **OK** button.



   The *ViewProducts* and *UpdateProduct* procedures appear in the Application Tree. They appear "disconnected" from the rest of the procedures because no other procedure calls them (yet). We'll do that next.



Working with Embed Points

The Clarion for Windows templates allow you to add your own customized code to many predefined points inside the standard code that the template generates. It's a very efficient way to achieve maximum code reusability and flexibility.

The point at which your code is inserted is called an *Embed Point*. Embed points are at all the standard events for the window and each control and many other logical positions within the generated code.

In this example, you add embedded source code—using a Code template that will write the actual source for you—at the points where the end user chooses the **View ➤ Products** menu item, and at the point where the end user presses the Products button on the application toolbar.

❑ Name the procedure to call for the **View ➤ Products** command.

1. RIGHT-CLICK on the *Main* procedure in the Application Tree.

   This is the procedure containing the menu and toolbar.

2. Choose **Embeds** from the popup menu.

   The **Embedded Source** dialog appears, which allows access to the embed points. You can also get here from the **Embeds** button on the **Procedure Properties** window, but the popup menu is quicker.

3. Press the **Contract All** button.

   This will make it easier to locate the specific embed point you need.

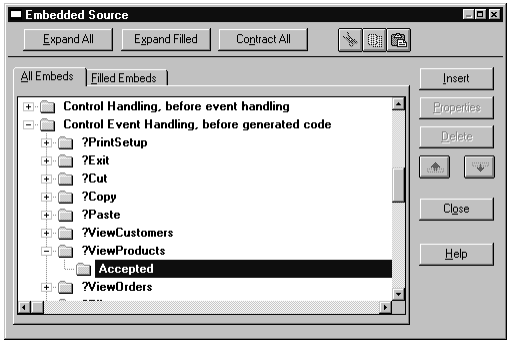
4. Locate the **Control Event Handling, before generated code** folder, then CLICK on its + sign to expand its contents.

   The menu selection is considered a control.

5. Locate the **?ViewProducts** folder, then CLICK on it to expand it.

6. Highlight **Accepted** then press the **Insert** button.

   The “Accepted” event for this menu selection marks the point in the generated code that executes when the user chooses the menu command.

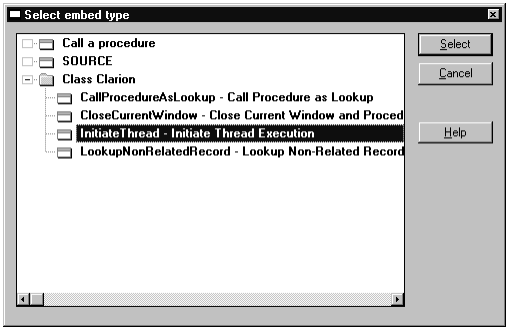


The **Select embed type** dialog appears to list all your options for embedding code. You may simply **Call a Procedure**, write your own **SOURCE** in the Text Editor, or use a Code template to write the source code for you.

7. Highlight the **InitiateThread** Code template, then press the **Select** button.

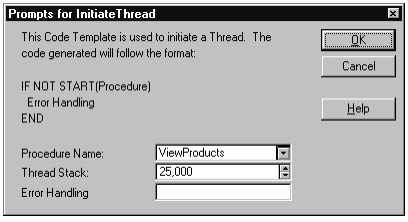
   A Code template usually provides just a few prompts and instructions on its use. It gathers the information it needs from you to write its executable code, which it then inserts into the standard generated code produced by the Procedure template directly into this embed point.

   This Code template is designed to start a new execution thread by calling the procedure you name using the START function.



8. Choose *ViewProducts* from the **Procedure Name** dropdown list.

   This names the procedure to call when the user chooses the menu item. This is the name of the procedure you previously copied.



9. Press the **OK** button.

❑ Name the procedure to call for the **Products** tool bar button.

At this point, you could go through the same process you just finished and use the **InitiateThread** Code template to call the *ViewProducts* procedure when the user presses the **Product** button in your program. However, now that you've done it once there's an easier way to write this code; just Copy and Paste it from one embed point to another!

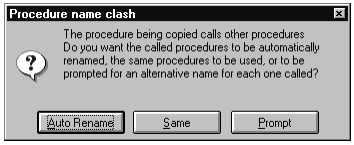
1. CLICK on the **Copy** button (middle button of the button group to the right of the **Contract All** button).

   The Code template you just added should still be highlighted, so this will copy it to the Windows clipboard.

2. Locate the *?ProductsButton* folder (in the same *Control Event Handling, before generated code* folder you are already in), then CLICK on it to expand it.

3. Highlight *Accepted* then press the **Paste** button (far right button of the button group to the right of the **Contract All** button).

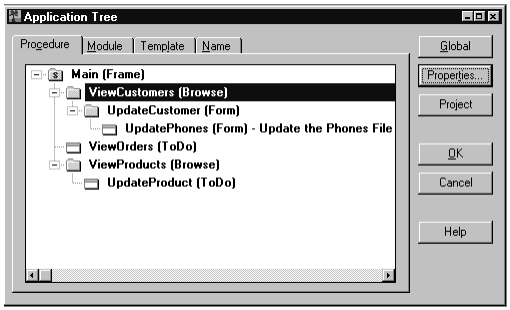
   The **Procedure name clash** dialog appears again to warn you that you've already called this procedure once.



4. Press the **Same** button.

   Now this embed point will generate the same code as the previous one.

4. Press the **Close** button.



   The *ViewProducts* procedure now "connects" beneath the *Main* procedure.

MODIFYING THE COPIED PROCEDURES

Now you can customize the copied procedures to reference the Products file.

❑ Change the file for the browse list control.

1. RIGHT-CLICK the *ViewProducts* procedure and choose **Window** from the popup menu.

2. RIGHT-CLICK the listbox control and choose **List Box Format...** from the popup menu.

3. In the List Box Formatter, press the **Delete** button repeatedly until all the fields are removed.
4. Press the **Populate** button.
5. Highlight the *Customer* file in the **Files** list, then press the **Delete** button.
6. Highlight the "ToDo" which replaces the *Customer* file, then press the **Insert** button.
7. Highlight the *Products* file then press the **Select** button.

   The **Select Field** dialog now lists the correct file and fields for this procedure.
8. Press the **Key** button, then select *KeyProdNumber* from the **Change Access Key** dialog.

   The **Select Field** dialog now lists the correct file and fields for this procedure.

❏ Re-populate the fields.

1. Highlight *PRD:ProdNumber* in the **Fields** list, then press the **Select** button.
2. Press the **Properties** button then select the **Field** tab.
3. Check the **Right Border** and **Resizeable** boxes.
4. Press the **OK** button.
5. Press the **Populate** button.
6. Highlight *PRD:ProdDesc* in the **Fields** list, then press the **Select** button.
7. Press the **Populate** button.
8. Highlight *PRD:ProdAmount* in the **Fields** list, then press the **Select** button.
9. Press the **Populate** button.
10. Highlight *PRD:TaxRate* in the **Fields** list, then press the **Select** button.
11. Press the **OK** button to close the List Box Formatter.

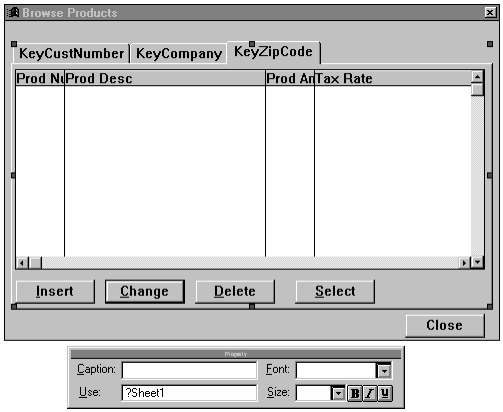
❏ Change the name of the window.

1. CLICK on the sample window caption bar.
2. Type *Browse Products* in the **Caption** field of the floating **Propertybox** toolbox, then press TAB.

❑ Remove all the tabs.

1. CLICK immediately to the right of the *KeyZipCode* tab to select the entire property sheet.

   To be sure that you have CLICKED in the right place, look at the floating **Propertybox** and make sure that its **Use** field displays *?Sheet1*. If it does not, try again until it does.



2. Press DELETE on the keyboard.

   All the tabs disappear.

3. Choose **Exit!** on the menu to close the Window Formatter (save your changes).

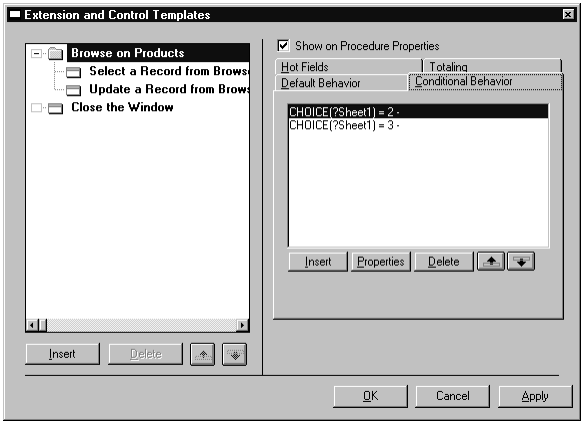
❑ Clean up the alternate sort orders.

1. RIGHT-CLICK the *ViewProducts* procedure and choose **Extensions** from the popup menu.

   The **Extension and Control Templates** dialog appears. This dialog lists all the **Actions** prompts for all the Control templates that the procedure uses. It also allows you to add and maintain Extension templates to the procedure.

   Extension templates are very similar to Control templates, in that, they add specific functionality to the procedure, but an Extension template's functionality is not directly associated with any control(s) on the window.

2. Highlight **Browse on Products** then select the **Conditional Behavior** tab.



3. Press the **Delete** button twice.

   This removes the two conditional expressions we entered for the *ViewCustomers* procedure.

4. Press the **OK** button.

5. Choose **File ➤ Save**, or press the *Save* button on the toolbar.

Creating the Form Procedure

When you renamed the reference to the *UpdateCustomer* procedure while copying *ViewCustomer* to *ViewProducts*, it made the *UpdateProduct* procedure a "ToDo" procedure. Therefore, we need to create a form to update the Products file.

❏ Select the procedure type for UpdateProduct

1. Highlight *UpdateProduct* then, press the **Properties** button.
2. Highlight *Form*, uncheck the **Use Procedure Wizard** dialog, then press the **Select** button.
3. Press the **Files** button in the **Procedure Properties** dialog.
4. Highlight the "ToDo" file, then press the **Insert** button.
5. Highlight the *Products* file then press the **Select** button.
6. Press the **OK** button to return to the **Procedure Properties** dialog.
7. Press the **Window** button to design your form.

❏ Populate the fields.

1. Choose **Options ➤ Show Fieldbox** to display the floating **Fieldbox** toolbox.

2. DOUBLE-CLICK on *ProdNumber* in the **Fieldbox** toolbox.

   This automatically places both the prompt and entry box for the field near the top left corner of the window.

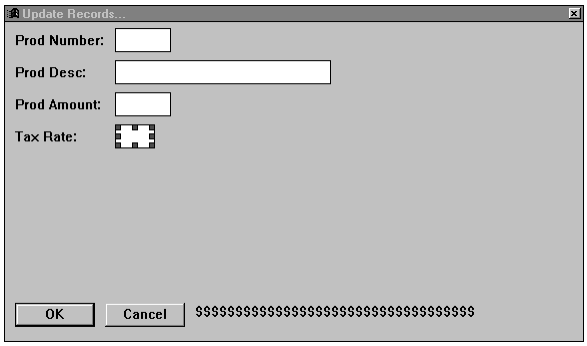
3. DOUBLE-CLICK on *ProdDesc* in the **Fieldbox** toolbox.

   This automatically places both the prompt and entry box for the field immediately below the last field that was placed.

4. DOUBLE-CLICK on *ProdAmount* in the **Fieldbox** toolbox.

5. DOUBLE-CLICK on *TaxRate* in the **Fieldbox** toolbox.

   The form window now looks something like this:



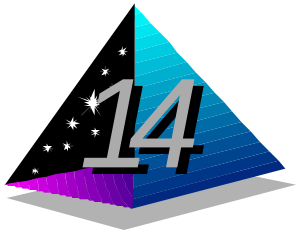
❑ Change the form window caption, exit the Window Formatter, and save your work.

1. CLICK on the caption bar of the sample window.
2. Type *Product Form* in the **Caption** field of the floating **Propertybox** toolbox, then press TAB.
3. Choose **Exit!** on the menu to close the Window Formatter (save your changes).
4. Press the **OK** button in the **Procedure Properties** dialog to close it.
5. Choose **File ➤ Save**, or press the *Save* button on the toolbar to save your work.

   The form window is done.

CREATING THE VIEWORDERS PROCEDURE

Contents



For the *ViewOrders* procedure, you'll create a window with two synchronized scrolling list boxes. One displays the contents of the Orders file, and the other displays the related records in the Detail file.

You'll use a generic window procedure, and populate it using Control templates. Control templates generate all the source code for creating *and maintaining* a control. In this case, placing a *BrowseBox* Control template allows the Application Generator to produce the code that opens the files and puts the necessary data into QUEUE structures (see the *Language Reference*) which hold the data for display in the list box.

CREATING THE PROCEDURE

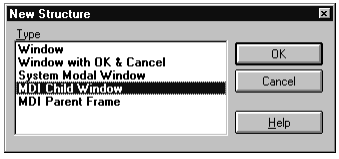
❑ Select the procedure type for the ViewOrders procedure:

1. Highlight *ViewOrders* then press the **Properties** button.
2. Highlight *Window* in the **Select Procedure Type** dialog then press the **Select** button.

❑ Edit the Window procedure.

1. In the **Procedure Properties** dialog, press the **Window** button.

   The **New Structure** dialog appears. The generic window procedure is like a blank slate in which you define your own window. Since the procedure has no predefined window, you choose the type of window will provide your starting point. In this case, you need an MDI child window.



2. Highlight *MDI Child Window*, then press the **OK** button.

   The Window Formatter appears.

3. Resize the window, making it about twice its original size.

4. Type *Orders* in the **Caption** field of the floating **Propertybox** toolbox, then press TAB.

Placing the First Control Template.

1. Choose **Populate ➤ Control Template**, or CLICK on the *Control Template* tool in the floating **Controls** toolbox (last tool icon on the right, bottom row).

   The **Select Control Template** dialog appears.

2. Highlight the *BrowseBox* Control template, then press the **Select** button.

   The cursor changes to a crosshair and "little book."

3. CLICK near the upper left corner of the sample window.

   The List Box Formatter appears, ready for you to choose fields.

❑ Place the Orders file fields in the List Box Formatter.

1. Press the **Populate** button.

2. Highlight the "ToDo" item below the **File-Browsing List Box** and press the **Insert** button.

3. Highlight the *Orders* file in the **Insert File** dialog, then press the **Select** button.

4. Press the **Key** button and select *KeyOrderNumber* from the **Change Access Key** dialog.

5. Highlight *ORD:CustNumber* in the **Fields** list, then press the **Select** button.

6. Press the **Populate** button.

7. Highlight *ORD:OrderNumber* in the **Fields** list, then press the **Select** button.

8. Press the **Populate** button.

9. Highlight *ORD:InvoiceAmount* in the **Fields** list, then press the **Select** button.

10. Press the **Populate** button.

11. Highlight *ORD:OrderDate* from the **Fields** list, then press the **Select** button.

12. Press the **Populate** button.

13. Highlight *ORD:OrderNote* in the **Fields** list, then press the **Select** button.
14. Press the **OK** button to close the List Box Formatter.
15. Resize the browse list box control by dragging the handle in the middle of the right side to the right, making it wider (almost as wide as the window).

❏ Format the list box appearance.

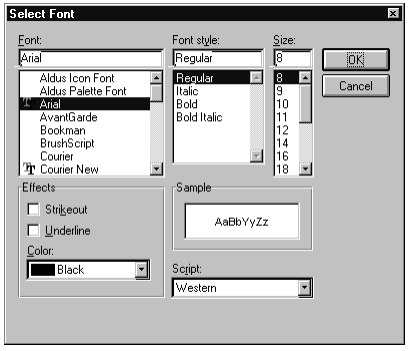
1. RIGHT-CLICK the list box, and select **Properties** from the popup menu.
2. Select the **Extra** tab and check the **Vertical** and **Horizontal** boxes in the **List Properties** dialog.

   This adds vertical and horizontal scroll bars to the list.
3. Press the **Font** button.

   Because one field (the description field) is long, you can specify that the list box should use a smaller font, displaying more information without requiring the end user to scroll.
4. Choose a font (your choice), and set the size to 8 points.

   See the *User's Guide* for tips on topics like choosing the right fonts for controls. In general, you want to stick with the fonts that ship with Windows; otherwise, you can't be sure your end user has the same font on their system.

   The illustration below sets the font to Arial, which is a font that ships with Windows.



5. Press **OK** to close the **Select Font** dialog.
6. Press **OK** to close the **List Properties** dialog.

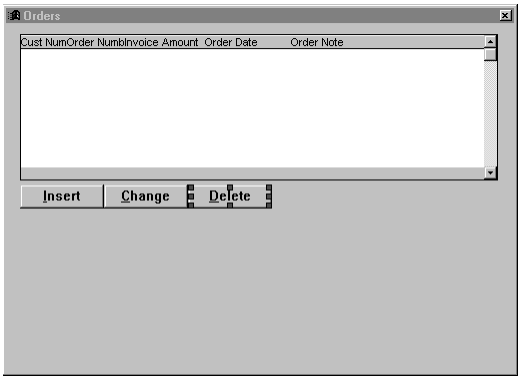
Adding the Browse Update Buttons Template

Next add the standard **Insert**, **Change** and **Delete** buttons for the top browse list box control. We'll later add a form procedure for adding or editing an order.

1. Choose **Populate ➤ Control Template**, or CLICK on the Control Template tool in the Controls toolbox (last tool icon on the right, bottom row).
2. In the **Select Control Template** dialog, highlight the *BrowseUpdateButtons* Control template, then press the **Select** button.
3. CLICK below the left edge of the list box.

   The **Insert**, **Change**, and **Delete** buttons all appear together.

   At this point the window should look something like this:



❑ Name the Update Procedure.

1. RIGHT-CLICK on the **Delete** button, then choose **Actions** from the popup menu.
2. Type *UpdateOrder* in the **Update Procedure** box.

   This names the procedure, in the same way that you named the Update procedure for the Customer browse in its **Procedure Properties** dialog.

   Naming the Update Procedure for one button in the Control template names it for all three.
3. Press the **OK** button.

Placing the Second Browse List Box

Next, place a list box with the contents of the Detail file. This updates automatically when the end user changes the selection in the top list box.

1. Choose **Populate ➤ Control Template**, or CLICK on the Control Template tool in the Controls toolbox (last tool icon on the right, bottom row).
2. Highlight the *BrowseBox* Control template, then press the **Select** button.
3. CLICK directly below the **Insert** button you placed before.

   The List Box Formatter appears, ready for you to choose fields.

❑ Place the Detail file fields in the List Box Formatter.

1. Press the **Populate** button.
2. Highlight the "ToDo" item below the *second* **File-Browsing List Box** and press the **Insert** button.
3. Highlight the *Detail* file in the **Insert File** dialog, then press the **Select** button.
4. Press the **Key** button.
5. Highlight *KeyOrderNumber* in the **Change Access Key** dialog, then press the **Select** button.
6. Highlight *DTL:OrderNumber* in the **Fields** list, then press the **Select** button.
7. Press the **Populate** button.
8. Highlight *DTL:ProdNumber* in the **Fields** list, then press the **Select** button.
9. Press the **Populate** button.
10. Highlight *DTL:Quantity* in the **Fields** list, then press the **Select** button.
11. Press the **Populate** button.
12. Highlight *DTL:ProdAmount* in the **Fields** list, then press the **Select** button.
13. Press the **Populate** button.
14. Highlight *DTL:TaxRate* in the **Fields** list, then press the **Select** button.
15. Press the **OK** button to close the List Box Formatter.

16. Resize the browse list box control by dragging the handle in the middle of the right side to the right, making it wider (about half as wide as the window).

❑ Set up the Control template's functionality by specifying the Range Limits.

1. RIGHT-CLICK on the list box you just placed and select **Actions** from the popup menu.
2. Press the ellipsis (...) button for the **Range Limit Field**.
3. Highlight the *DTL:OrderNumber* field in the **Components** list, then press the **Select** button.
4. Choose **File Relationship** from the **Range Limit Type** drop down list.
5. Press the ellipsis (...) button in the **Related File**.
6. Highlight the *Orders* file in the **Select File** list, then press the **Select** button.

   These last four steps limit the range of records displayed in the second list box to only those Detail records related to the currently highlighted record in the Orders file's list box.

   This tells the second control template to use the file relationship defined in the data dictionary to synchronize the bottom list to the top.

❑ Format the list box appearance.

1. Select the **Extra** tab.
2. Check the **Vertical** and **Horizontal** boxes.

   This adds horizontal and vertical scroll bars to the list box.
3. Press the **Font** button.

   Although there are no "long" fields in this list box, it will look better if you match the font to the same font used in the top list box.
4. Choose a font (your choice), and set the size to 8 points.
5. Press **OK** to close the **Select Font** dialog.
6. Press **OK** to close the **List Properties** dialog.

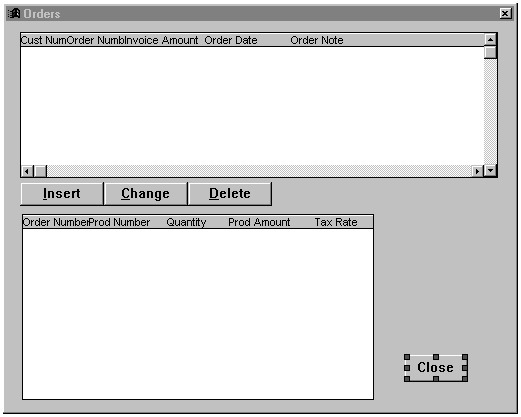
Adding the Close Button Control Template

Finally, you can add a **Close** button that closes the window.

1. Choose **Populate ➤ Control Template**, or CLICK on the Control Template tool in the **Controls** toolbox (last tool icon on the right, bottom row).
2. Select the *CloseButton* Control template, then press the **Select** button.
3. CLICK in the lower right corner of the window.

   At this point, your window may look like this illustration.

   It may be bigger than the sample area in the Window Formatter, but should not be as big as the desktop:

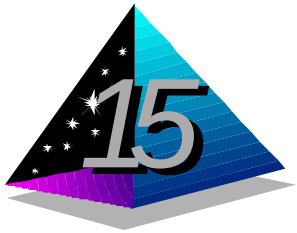


❑ Close the Window Formatter and Save the Application.

1. Choose **Exit!** on the Window Formatter menu to close the Window Formatter.
2. Press the **OK** button in the **Procedure Properties** dialog to close it.
3. Choose **File ➤ Save**, or press the *Save* button on the tool bar to save your work.

CREATING THE UPDATEORDER FORM

Contents



For the Order Update form, we'll place the fields from the Order file on an update form, perform an automatic lookup to the Customer file, add a BrowseBox Control template showing the related detail items, calculate each line item detail, then calculate the order total.

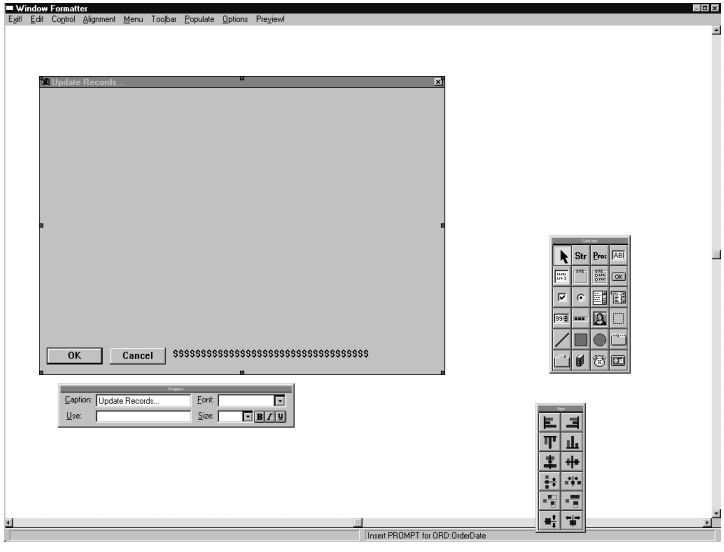
Set Up the Basic Form

❑ Select the procedure type for the UpdateOrder procedure:

1. Highlight *UpdateOrder* in the **Application Tree** dialog, then press the **Properties** button.
2. Highlight *Form* in the **Select Procedure Type** dialog, uncheck the **Use Procedure Wizard** box, then press the **Select** button.

❑ Edit the Window procedure.

1. Press the **Window** button then make the window taller by DRAGGING its top middle handle (use the Window Formatter's vertical scroll bar to move the viewing area, if necessary).

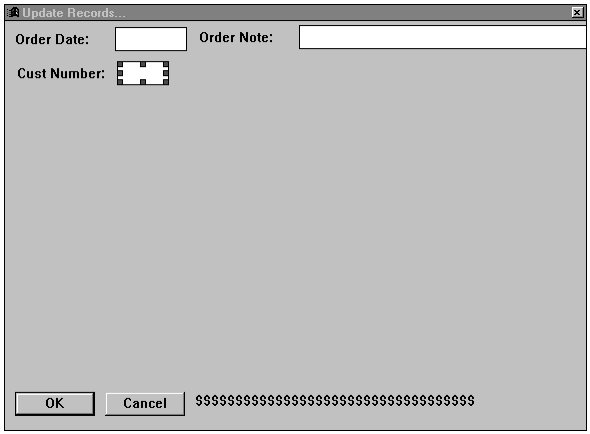


❑ Place the entry controls for the Orders file.

1. Choose **Populate ➤ Multiple Fields**, or select the *Dictionary Field* tool from the floating **Controls** toolbox.
2. In the **Select Field** dialog, highlight the "ToDo" item under the **Update Record on Disk**, then press the **Insert** button.
3. Select the *Orders* file from the **Insert File** list.
4. Highlight *ORD:OrderDate*, then press the **Select** button.
5. CLICK near the top left of the window.

   This places both the field and its prompt. The **Select Field** dialog immediately reappears, ready for the next field.
6. Highlight *ORD:OrderNote*, then press the **Select** button.
7. CLICK just to the right of the entry box placed for the date.
8. Highlight *ORD:CustNumber*, then press the **Select** button.
9. CLICK below the date field's prompt.
10. Press the **Cancel** button in the **Select Field** dialog to exit the multi-populate mode.

    The form window now looks something like this illustration.



❑ Add a lookup procedure call into the customer list.

1. RIGHT-CLICK on the *ORD:CustNumber* control and select **Actions** from the popup menu.

   The standard actions for any entry control allow you to perform data entry validation against a record in another file, either when the control is Selected (before the user can enter data) or when the control is Accepted (after the user has entered data).

2. In the **When the Control is Accepted** group box, press the ellipsis button (...) for the **Lookup Key** entry box.
3. Highlight the *Orders* file in the **Select Key** dialog, then press the **Insert** button.
4. Highlight the *Customer* file in the **Files** list, and press the **Select** button.

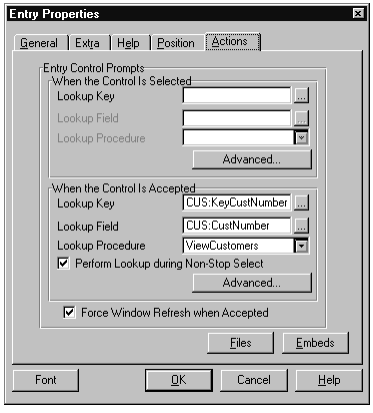
   These last two steps add the *Customer* file to the procedure's File Schematic as an automatic lookup from the *Orders* file. This is based on the file relationship set up in the data dictionary.
5. Highlight *CUS:KeyCustNumber* in the **Select Key** dialog, then press the **Select** button.

   This makes *CUS:KeyCustNumber* the key that will be used to attempt to get a matching valid record from the Customer file for the value the user enters into this control.
6. Press the ellipsis button (...) for the **Lookup Field** entry box.
7. Highlight *CUS:CustNumber* from the **Select Component** list, then press the **Select** button.

   This makes *CUS:CustNumber* the field that must contain the matching value to the value the user enters into this control.
8. Choose the **ViewCustomers** procedure from the **Lookup Procedure** drop-down list.

   This calls the *ViewCustomers* procedure when the user enters an invalid value for the customer number, allowing the end user to select from a list of customers.
8. Check the **Perform Lookup during Non-Stop Select** and **Force Window Refresh when Accepted** boxes.

   These ensure that the data displayed on screen is always valid and current.



9. Press the **Embeds** button in the **Prompts** dialog.
10. Highlight the **Selected** event under **Control Event Handling, after generated code** then press the **Insert** button.
11. Highlight *SOURCE* then press the **Select** button to call the Text Editor.
12. Type in the following code:

```
?ORD:CustNumber{PROP:Touched} = TRUE
```

    This statement uses the Clarion language Property Assignment syntax (see Appendix C in the *Language Reference*) to ensure that an *Accepted* event is always generated for the control (whether the user enters data or not) which forces the user to enter valid data.
13. Choose **Exit!** (and save) to return to the **Embedded Source** dialog.
14. Press the **Close** button to return to the **Entry Properties** dialog, then press the **OK** button.

❏ Add a "display-only" control.

1. Choose **Control ➤ String**, or CLICK on the **String** tool in the floating **Controls** toolbox (the icon just right of the "big arrow" on the top row).
2. CLICK to the right of the customer number entry box you placed before.
3. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
4. Check the **Variable String** box.

    This specifies the control will display data from a variable, not just a string constant. The **Select Field** dialog automatically appears.
5. Highlight the *Customer* file in the **Files** list, then select *CUS:Company* from the **Fields** list and press the **Select** button.
6. Press the **OK** button to close the **String Properties** dialog.

Placing the Detail File's Control Templates

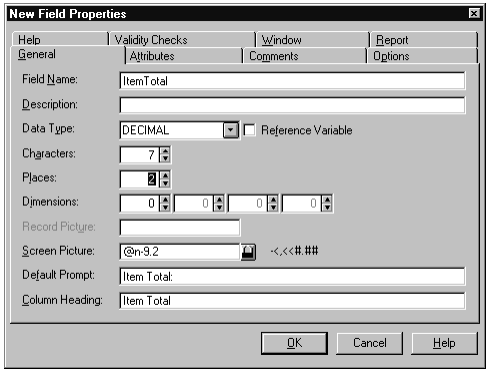
The next key element in this window is a browse list box control, synchronized to the Order Number of this form, showing the related records in the Detail file.

❏ Add a Detail list.

1. Choose **Populate ➤ Control Template**, or CLICK on the Control Template tool in the floating **Controls** toolbox (last tool icon on the right, bottom row).
2. Highlight *BrowseBox*, then press the **Select** button.

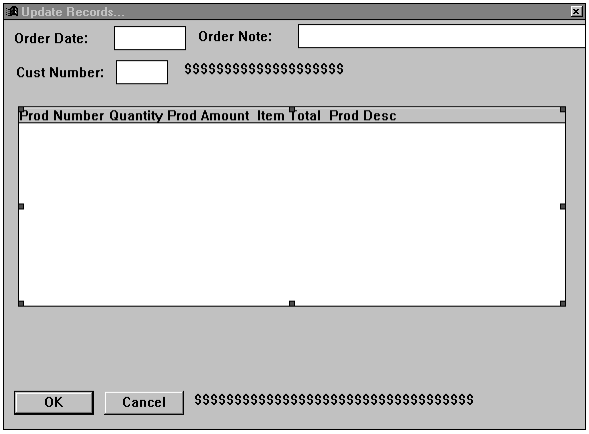
3. CLICK below the customer number entry box you placed before.
4. Press the **Populate** button.
5. Highlight the "ToDo" item below the **File-Browsing List Box** and press the **Insert** button.
6. Select the *Detail* file from the **Insert File** dialog, then press the **Select** button.
7. Press the **Key** button.
8. Highlight *KeyOrderNumber* in the **Change Access Key** dialog, then press the **Select** button.
9. Highlight *DTL:ProdNumber* in the **Fields** list, then press the **Select** button.
10. Press the **Populate** button.
11. Highlight *DTL:Quantity* in the **Fields** list, then press the **Select** button.
12. Press the **Populate** button.
13. Highlight *DTL:ProdAmount* in the **Fields** list, then press the **Select** button.
14. Press the **Populate** button.
15. Highlight *LOCAL DATA UpdateOrder* in the **Files** list, then press the **New** button.

    This allows you to add a local variable without going back to the **Procedure Properties** window and pressing the **Data** button. This variable will be used to display the total price for each line item.
16. Type *ItemTotal* in the **Name** field.
17. Select *DECIMAL* from the **Type** dropdown list.
18. Type *7* in the **Characters** field.
19. Type *2* in the **Places** field then press the **OK** button.



20. Press the **Populate** button.
21. Highlight the *Detail* file item below the **File-Browsing List Box** and press the **Insert** button.
22. Select the *Products* file from the **Insert File** dialog, then press the **Select** button.

    This adds the *Products* file to the Control template's file schematic as a "lookup" file. The related record from the *Products* file is automatically retrieved for you so you can display the product description in the list box.
23. Highlight *PRD:ProdDesc* in the **Fields** list, then press the **Select** button.
24. Resize the columns then press the **OK** button to close the List Box Formatter.



❏ Synchronize the browse list box to the order number of the current record.

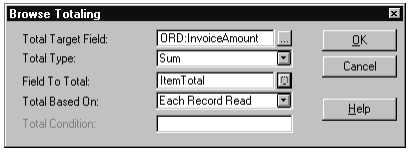
1. RIGHT-CLICK the list box you just placed, and select **Actions** from the popup menu.
3. Press the ellipsis (...) button for the **Range Limit Field**.
4. Highlight the *DTL:OrderNumber* field in the **Components** list, then press the **Select** button.
5. Choose **File Relationship** from the **Range Limit Type** drop down list.
6. Type *Orders* in the **Related File** field.

❏ Add a total calculation to the browse list box for the order invoice amount.

1. Select the **Totaling** tab.
2. Press the **Insert** button.
3. Press the ellipsis (...) button for the **Total Target Field**.
4. Highlight the *Orders* file in the **Files** list, select *ORD:InvoiceAmount* from the **Fields** list, then press the **Select** button.

   This is the field that will receive the result of the total calculation.
5. Choose **Sum** from the **Total Type** drop down list.
6. Press the ellipsis (...) button for the **Field to Total**.
7. Highlight *LOCAL DATA UpdateOrder* in the **Files** list, select *ItemTotal* from the **Fields** list, then press the **Select** button.

   This is the field whose contents will be used in the total calculation.
8. Choose *Each Record Read* from the **Total Based On** dropdown list.



9. Press **OK** to close the **Browse Totaling** dialog.

❏ Add horizontal and vertical scroll bars.

1. Select the **Extra** tab.
2. Check the **Horizontal** and **Vertical** boxes.
3. Press **OK** to close the **List Properties** dialog.

❏ Add the standard **Insert**, **Change** and **Delete** buttons.

1. Choose **Populate ➤ Control Template**, or CLICK on the Control Template tool in the **Controls** toolbox (last tool icon on the right, bottom row).
2. In the **Select Control Template** dialog, select the *BrowseUpdateButtons* control template, then press the **Select** button.

   The cursor changes to a crosshair and “little book.”
3. CLICK below the list box.

   The **Insert**, **Change**, and **Delete** buttons all appear together.

4. RIGHT-CLICK on the **Delete** button, then choose **Actions** from the popup menu.
5. Type *UpdateDetail* in the **Update Procedure** entry box.

   Naming the Update Procedure for one button in the Control template names it for all three.
6. Press the **OK** button.

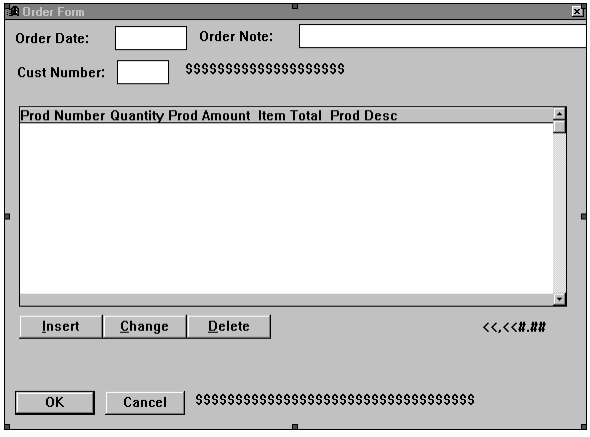
❏ Add a "display-only" control for the invoice total.

1. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.
2. CLICK below the bottom right corner of the list box.
3. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
4. Check the **Variable String** box.

   This specifies the control will display data from a variable, not just a string constant. The **Select Field** dialog automatically appears.
5. Highlight the *Orders* file in the **Files** list, then select *ORD:InvoiceAmount* from the **Fields** list and press the **Select** button.
6. Change the **Picture** field to *@N9.2* then press the **OK** button.

❏ Change the form window caption and exit the Window Formatter.

1. CLICK on the caption bar of the sample window.
2. Type *Order Form* in the **Caption** field in the floating **Propertybox** toolbox, then press TAB.



3. Choose **Exit!** to close the Window Formatter.

MAKING IT WORK

To make the ItemTotal calculate the correct amount for each Detail record in the browse list box, you need to add a Formula to the procedure. This will also allow the browse totaling to correctly place the invoice total in the ORD:InvoiceAmount field.

❑ Add the Formula.

1. Press the **Formulas** button in the **Procedure Properties** dialog.

   The **Formula Editor** dialog appears.

2. Type *Item Total Formula* in the **Name** field.

3. Press the ellipsis (...) button in the **Class** Field.

4. Highlight *Format Browse* in the **Template Classes** list, then press the **OK** button.

   The **Class** field defines the logical position within the generated source code at which the formula is calculated. The *Format Browse* class tells the *BrowseBox* Control template to perform the calculation each time it formats a record for display in the list box.

6. Press the ellipsis (...) button in the **Result** Field.

7. Highlight *LOCAL DATA UpdateOrder* in the **Files** list, select *ItemTotal* from the **Fields** list, then press the **Select** button.

   This names the field that will receive the result of the calculation. This is the field we defined earlier through the List Box Formatter.

8. Press the **Data** button in the **Operands** group.

9. Highlight the *Detail* file in the **Files** list, select *DTL:Quantity* from the **Fields** list, then press the **Select** button.

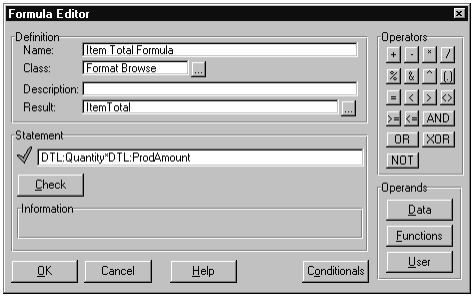
   This places the *DTL:Quantity* field in the **Statement** field for you. The **Statement** field contains the expression being built, and you can also type directly into it to build the expression, if you wish.

10. Press the ***** button in the **Operators** group.

11. Press the **Data** button in the **Operands** group.

12. Highlight the *Detail* file in the **Files** list, select *DTL:ProdAmount* from the **Fields** list, then press the **Select** button.

13. Press the **Check** button to check the expression's syntax.



14. Press the **OK** button to close the Formula Editor.

    The **Formulas** dialog appears. Since there were no pre-defined formulas, you did not see this window when you first pressed the **Formulas** button on the **Procedure Properties** window. Now that there is one, the next time you press the **Formulas** button you will see this dialog first.

15. Press the **OK** button to close the **Formulas** dialog.

16. Press the **OK** button in the **Procedure Properties** dialog to close it.

17. Choose **File ➤ Save**, or press the *Save* button on the toolbar to save your work.

Add the UpdateDetail Form Procedure

Now we need to create the UpdateDetail Form procedure to maintain the Detail file records.

❑ Select the procedure type for the UpdateDetail procedure:

1. Highlight *UpdateDetail* in the **Application Tree** dialog, then press the **Properties** button.
2. Highlight *Form* in the **Select Procedure Type** dialog, uncheck the **Use Procedure Wizard** box, then press the **Select** button.

❑ Place the entry controls for the Detail file.

1. Press the **Window** button.
2. Choose **Populate ➤ Multiple Fields**, or select the *Dictionary Field* tool from the **Controls** toolbox.
3. In the **Select Field** dialog, select the “ToDo” item under **Update Record on Disk** in the **Files** list, then press the **Insert** button.
4. Select the *Detail* file from the **Insert File** list.
5. Highlight *DTL:ProdNumber,* then press the **Select** button.

The **Select Field** dialog closes, and the cursor changes to a crosshair and a "little book."

6. CLICK near the top of the sample window, at the left.

   The **Select Field** dialog immediately reappears, ready for you to place the next field.

7. Highlight *DTL:Quantity*, then press the **Select** button.

8. CLICK below the last field you placed.

9. Press the **Cancel** button in the **Select a Field** dialog to exit multi-populate mode.

   These are the only entry fields required, since all the other fields in the *Detail* file can be filled in automatically. Of course, the value for the *DTL:OrderNumber* field comes from the *Orders* file, and the other fields will all come from the *Products* file.

❑ Add a lookup to the product list.

1. RIGHT-CLICK on the *DTL:ProdNumber* control and select **Actions** from the popup menu..

2. In the **When the Control is Accepted** group box, press the ellipsis button (...) for the **Lookup Key** entry box.

3. Select the *Detail* file from the **Select Key** dialog, then press the **Insert** button.

4. Select the *Products* file in the **Files** list, and press the **Select** button.

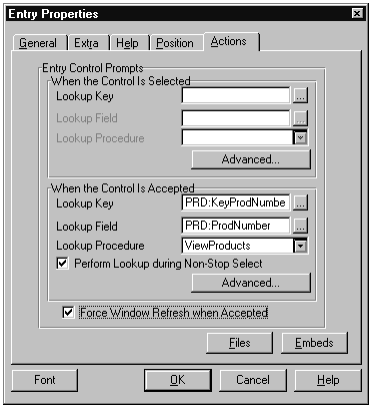
   These two steps add the *Products* file to the procedure as an automatic lookup file from the *Detail* file, based on the file relationship set up in the data dictionary.

5. HIghlight *PRD:KeyProdNumber* in the **Select Key** dialog, then press the **Select** button.

6. Press the ellipsis button (...) for the **Lookup Field** entry box.

7. HIghlight *PRD:ProdNumber* in the **Select Component** list, then press the **Select** button.

8. Choose the *ViewProducts* procedure from the **Lookup Procedure** dropdown list.



9. Press the **Embeds** button in the **Prompts** dialog.

10. Highlight the **Selected** event under **Control Event Handling, after generated code** then press the **Insert** button.

11. Highlight *SOURCE* then press the **Select** button to call the Text Editor.

12. Type in the following code:

```
?DTL:ProdNumber{PROP:Touched} = TRUE
```

    This code does the same thing that we did for the lookup field in the *UpdateOrder* procedure.

13. Choose **Exit!** to return to the **Embedded Source** dialog.

14. Highlight the **Accepted** event under **Control Event Handling, after generated code** then press the **Insert** button.

15. Highlight *SOURCE* then press the **Select** button.

16. Type in the following code:

```
DTL:Record :=: PRD:Record   !Deep assign like named variables
DISPLAY                     ! then display them on screen
```

    This uses Clarion's deep assignment operator (:=:, see the *Language Reference*) to automatically assign all the fields in the *Products* file to the fields in the *Detail* file that have the same names (while leaving all other fields alone). This means that, in this one statement, the *PRD:ProdNumber*, *PRD:ProdAmount*, and *PRD:TaxRate* values in the *Products* file record are copied into the *DTL:ProdNumber*, *DTL:ProdAmount*, and *DTL:TaxRate* fields.

17. Choose **Exit!** to return to the **Embedded Source** dialog.

18. Press the **Close** button to return to the **Entry Properties** dialog, then press the **OK** button to close that

❑ Add String constants.

1. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.

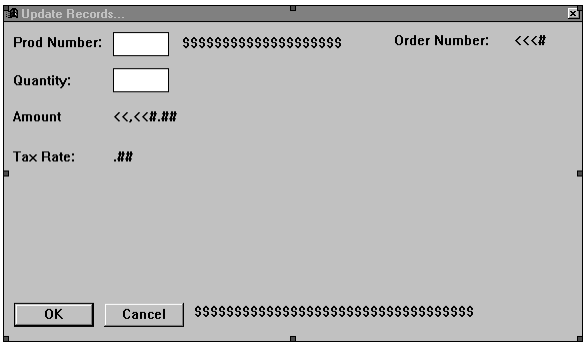
2. CLICK right of the first entry box you placed, near the right corner.
3. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
4. Type *Order Number:* in the **Parameter** field, then press the **OK** button.
5. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.
6. CLICK just below the *Quantity* prompt you placed.
7. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
8. Type *Amount:* in the **Parameter** field, then press the **OK** button.
9. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.
10. CLICK just below the last string you placed.
11. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
12. Type *Tax Rate:* in the **Parameter** field, then press the **OK** button.

❑ Add "display-only" controls.

1. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.
2. CLICK to the right of the *Order Number:* string you placed.
3. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
4. Check the **Variable String** box.

   This specifies the control will display data from a variable, not just a string constant. The **Select Field** dialog automatically appears.
5. Highlight the *Detail* file in the **Files** list, then select *DTL:OrderNumber* from the **Fields** list and press the **Select** button.
6. Change the **Picture** field to *@N4* then press the **OK** button to close the **String Properties** dialog.
7. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.
8. CLICK to the right of the *Product Amount:* string you placed.
9. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
10. Check the **Variable String** box.

11. Select *DTL:ProdAmount* from the **Fields** list and press the **Select** button.
12. Change the **Picture** field to *@N9.2* then press the **OK** button to close the **String Properties** dialog.
13. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.
14. CLICK to the right of the *Tax Rate:* string you placed.
15. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
16. Check the **Variable String** check box,
17. Select *DTL:TaxRate* from the **Fields** list and press the **Select** button.
18. Change the **Picture** field to *@N3.2* then press the **OK** button to close the **String Properties** dialog.
19. Choose **Control ➤ String**, or CLICK on the *String* tool in the **Controls** toolbox.
20. CLICK to the right of the *DTL:ProdNumber* entry box you placed.
21. RIGHT-CLICK the control you just placed, and select **Properties** from the popup menu.
22. Check the **Variable String** check box,
23. Highlight the *Products* file in the **Files** list, then select *PRD:ProdDesc* from the **Fields** list and press the **Select** button.
24. Press the **OK** button to close the **String Properties** dialog.
25. Choose **Exit!** to close the Window Formatter.
26. Press the **OK** button to close the **Procedure Properties** dialog.
27. Choose **File ➤ Save**, or press the *Save* button on the toolbar to save your work.



CREATING REPORTS

Contents

The last item to cover in this tutorial is adding reports. First, we'll create a simple customer list to introduce you to the Report Formatter. Then we'll create an Invoice Report to demonstrate how you can easily create Relational reports with multi-level group breaks, group totals, and page formatting.

A SIMPLE CUSTOMER LIST

❑ Add a menu item.

1. RIGHT-CLICK on the *Main* procedure in the **Application Tree** dialog and choose **Window** from the popup menu.
2. Choose **Menu ➤ Menu Editor** from the Window Formatter's menu.
3. Highlight the **P&rint Setup** item in the **Menu Editor** list.
4. Press the **Item** button.
5. Type *Print &Customer List* in the **Text** field then press TAB.

❑ Specify the new item's action.

1. Select the **Actions** tab.
2. Choose *Call a Procedure* from the **When Pressed** drop down list.
3. Type *CustReport* in the **Procedure Name** field.
4. Check the **Initiate Thread** box.

❑ Add a second menu item.

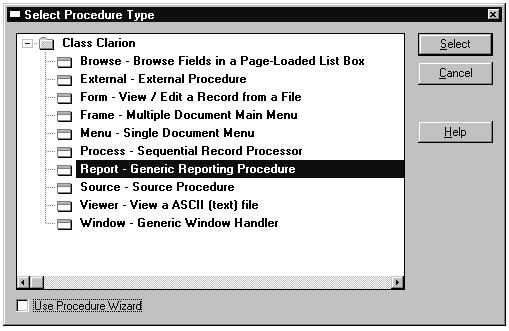
1. Press the **Item** button.
2. Type *Print &Invoices* in the **Text** field.
3. Select the **Actions** tab.
4. Choose *Call a Procedure* from the **When Pressed** drop down list.
5. Type *InvoiceReport* in the **Procedure Name** field.
6. Check the **Initiate Thread** box.

7. Press the **Close** button to close the **Menu Editor**.
8. Choose **Exit!** to close the Window Formatter (save your work).

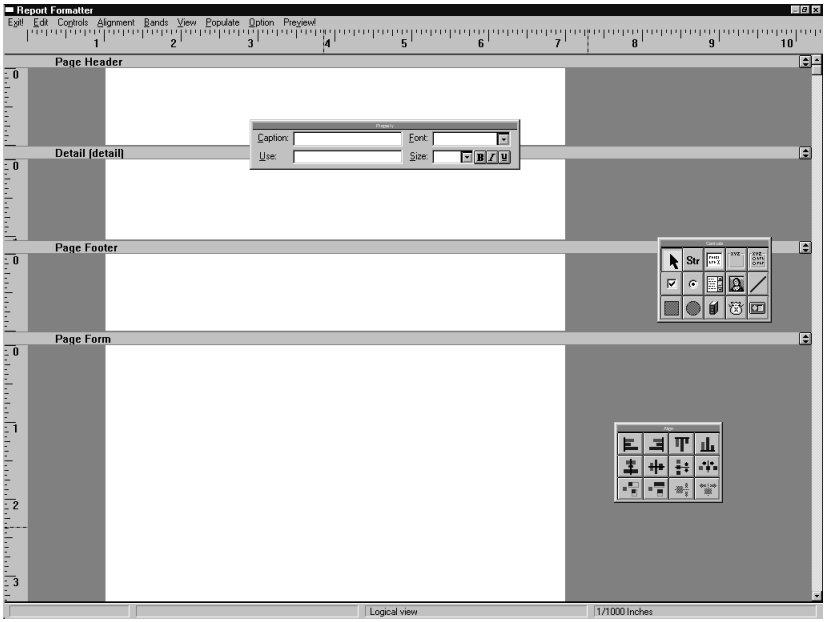
Creating a Report

Now you can create the first report, using the Report Formatter.

1. Highlight the *CustReport* procedure in the Application Tree.
2. Press the **Properties** button.
3. Highlight *Report* in the **Select Procedure Type** dialog, uncheck the **Use Procedure Wizard** box, then press the **Select** button.



4. Press the **Report** button in the **Procedure Properties** dialog.



The Report Formatter appears. Here you can visually edit the report and its controls. The Report Formatter represents the four basic parts of the REPORT data structure by showing the Page Header, Detail, Footer, and Form as four "bands." Each band is a single entity that prints all together. See the *User's Guide* chapter on *Using the Report Formatter* for more information on the parts of the report and how the print engine generates them.

For this report, you'll place page number controls in the header, then place the fields from the Customer file in the Detail band.

❏ Place a string constant.

1. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
2. CLICK at the top of the **Page Header** band.
3. RIGHT-CLICK on the control, then choose **Properties** from the popup menu.

   The **String Properties** dialog appears.
4. Type *Page Number:* in the **Parameter** field, then press the **OK** button.
5. Resize the control so that it's wide enough to hold the text, by DRAGGING its right handle.

❏ Place a control to print the Page Number.

6. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
7. CLICK to the right of the previously placed string.
8. RIGHT-CLICK on the control, then choose **Properties** from the popup menu.
9. Check the **Variable String** box.
10. Type *@n2* in the **Picture** field.

    This specifies a numeric picture for the control.
11. Type *?PageNumber* in the **Use** field.

    This makes the control's USE variable a field equate label. Printing a page number on a report does not require a variable.
12. Choose **Page No** from the **Total Type** drop down list.

    This tells the report to just put the page number in this control.
13. Press the **OK** button to close the **String Properties** dialog.

Populating the Detail

The Detail band prints once for each record in the report. For this procedure, you'll place the fields in a horizontal arrangement, which creates a columnar report at print time.

1. Choose **Populate ➤ Multiple Fields**, or CLICK on the *Dictionary Field* tool in the **Controls** tool box.
2. In the **File Schematic Definition** dialog, highlight the "ToDo" folder, then press the **Insert** button.
3. Select the *Customer* file from the **Insert File** dialog, then press the **Select** button.
4. Press the **Key** button.
5. Highlight *CUS:KeyCustNumber* in the **Change Access Key** dialog, then press the **Select** button.
6. Highlight *CUS:Company* in the **Fields** list and press the **Select** button.
7. CLICK inside the **Detail** band, near its top left corner.
8. Highlight *CUS:FirstName* in the **Fields** list and press the **Select** button.
9. CLICK inside the **Detail** band, just below the first control.
10. Highlight *CUS:LastName* in the **Fields** list and press the **Select** button.
11. CLICK inside the **Detail** band, to the right of the control you just placed.
12. Highlight *CUS:Address* in the **Fields** list and press the **Select** button.
13. CLICK inside the **Detail** band, below the *second* control you placed.

❏ Resize the Detail band:

At this point, you probably have very little room left in the Detail band, and need to make it longer.

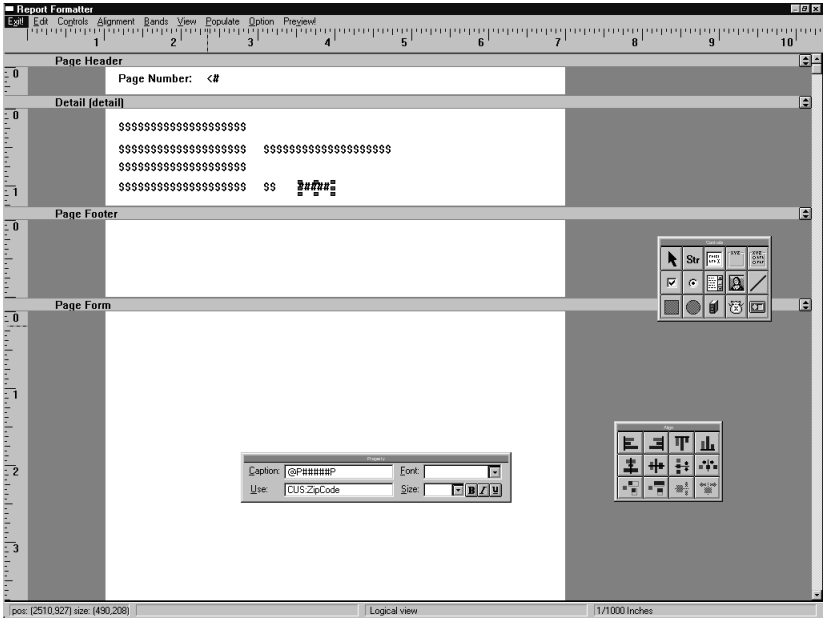
1. Press the **Cancel** button to close the **Select Field** dialog and exit multi-populate mode.
2. CLICK inside the **Detail** band, but not on one of the string controls.

   The Detail area's handles appear.
3. Resize the **Detail** band by dragging the middle handle on the bottom down—allow for enough room for about two more lines.

❏ Place the rest of the fields.

1. Choose **Populate ➤ Multiple Fields**, or CLICK on the *Dictionary Field* tool in the **Controls** tool box.
2. Highlight *CUS:City* in the **Fields** list, then press the **Select** button.
3. CLICK inside the **Detail** band, below the last control you placed.
4. Highlight *CUS:State* in the **Fields** list, then press the **Select** button.
5. CLICK inside the **Detail** band, to the right of the previously placed control.
6. Highlight *CUS:ZipCode* in the **Fields** list, then press the **Select** button.
7. CLICK inside the **Detail** band, to the right of the previously placed control.
8. Press the **Cancel** button to close the **Select Field** dialog and exit multi-populate mode.

   The Report Formatter should look something like this illustration:



   Notice that you have the same set of alignment tools in the Report Formatter that you have already used in the the Window Formatter.

❑ Select a base font for the Report.

1. Choose **Edit ➤ Report Properties** to set default attributes.
2. Press the **Font** button.
3. Select a font, style, and size to use as the base font for the report.

If you do not select a font for the report, it will print using the printer's default font. You should select a font that you know the user will have (the fonts that ship with Windows are usually safe).

4. Press the **OK** button to close the **Select Font** dialog.

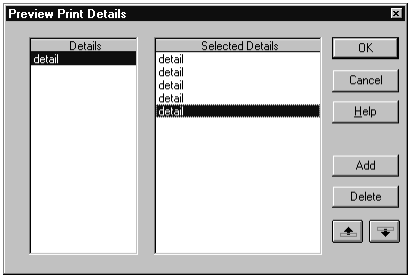
5. Press the **OK** button to close the **Report Properties** dialog.

❑ Preview the Report.

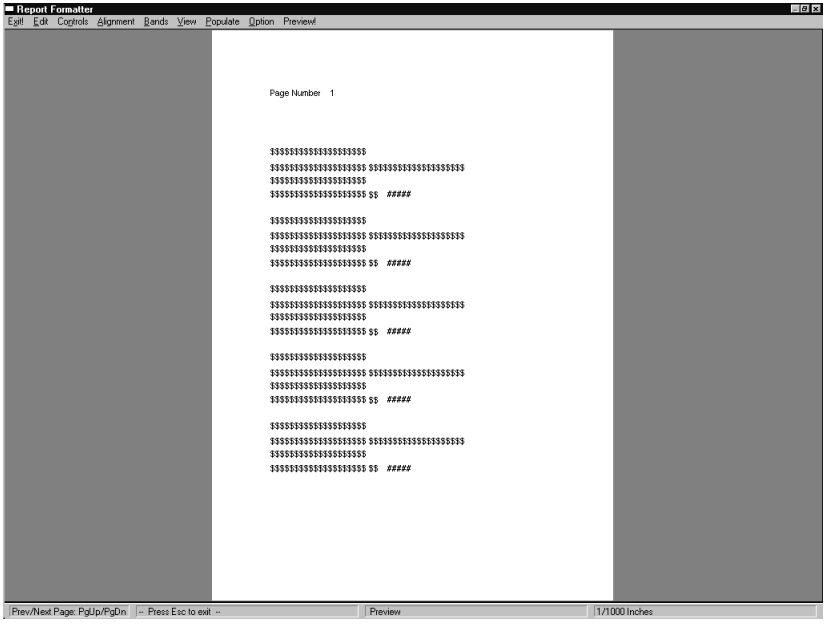
1. Choose **Preview!** to "visualize" how the printed page will appear.

2. Highlight *Detail* in the **Details** list then press the **Add** button several times.

   This populates the preview with some print bands to view. Because you can have many bands of various types within a single report, you have to select which to see before going to print preview. This way, the Report Formatter knows what to compose on the screen.



3. Press the **OK** button.



5. When done "previewing," choose **Band View!**.
6. Choose **Exit!** to return to the **Procedure Properties** dialog.
7. Press the **OK** button to close the **Procedure Properties** dialog.
8. Choose **File ➤ Save**, or press the *Save* button on the tool bar to save your work.

AN INVOICE REPORT

Next, we will create one of the most common types of reports. An invoice will make use of most of the files in the data dictionary, demonstrating how to create group breaks and totals. It will also show you how to control pagination based on group breaks.

Creating the Report

1. Highlight the *InvoiceReport* procedure.
2. Press the **Properties** button.
3. Highlight *Report* in the **Select Procedure Type** dialog, uncheck the **Use Procedure Wizard** box, then press the **Select** button.

   The **Procedure Properties** dialog appears.

❑ Specify the files for the Report.

1. Press the **Files** button in the **Procedure Properties** dialog.

   The **File Schematic Definition** dialog appears.
2. Highlight the "ToDo" folder, then press the **Insert** button.
3. Select the *Customer* file from the **Insert File** dialog, then press the **Select** button.
4. Press the **Key** button.
5. Highlight *CUS:KeyCustNumber* in the **Change Access Key** dialog, then press the **Select** button.

   The report will process all the *Customer* file records in *CustNumber* order.
6. Highlight the *Customer* file, then press the **Insert** button.
7. Select the *Orders* file from the **Insert File** dialog, then press the **Select** button.

   It will process all the *Orders* for each *Customer*.
8. Highlight the *Orders* file, then press the **Insert** button.

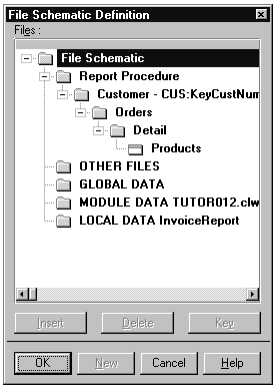
9. Select the *Detail* file from the **Insert File** dialog, then press the **Select** button.

   Each Order will print all the related *Detail* records.

8. Highlight the *Detail* file, then press the **Insert** button.

9. Select the *Products* file from the **Insert File** dialog, then press the **Select** button.

   Each *Detail* record will lookup the related *Products* file record. At this point, the File Schematic should look like this:



10. Press the **OK** button to return to the **Procedure Properties** dialog.

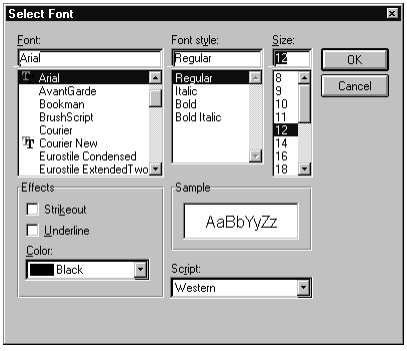
❑ Set the Report defaults.

1. Press the **Report** button.

2. Choose **Edit ➤ Report Properties** to set the report's default attributes.

3. Press the **Font** button.

4. Select a font, style, and size to use as the base font for the report.



If you do not select a font for the report, it will print using the printer's default font. You should select a font that you know the user will have (the fonts that ship with Windows are usually safe).

4. Press the **OK** button to close the **Select Font** dialog.
5. Press the **OK** button to close the **Report Properties** dialog.

Populating the Page Form Band

The Page Form band prints once for each page in the report. Its content is only composed once, when the report is opened. This makes it useful for constant information that will always be on every page of the report.

❑ Place a string constant.

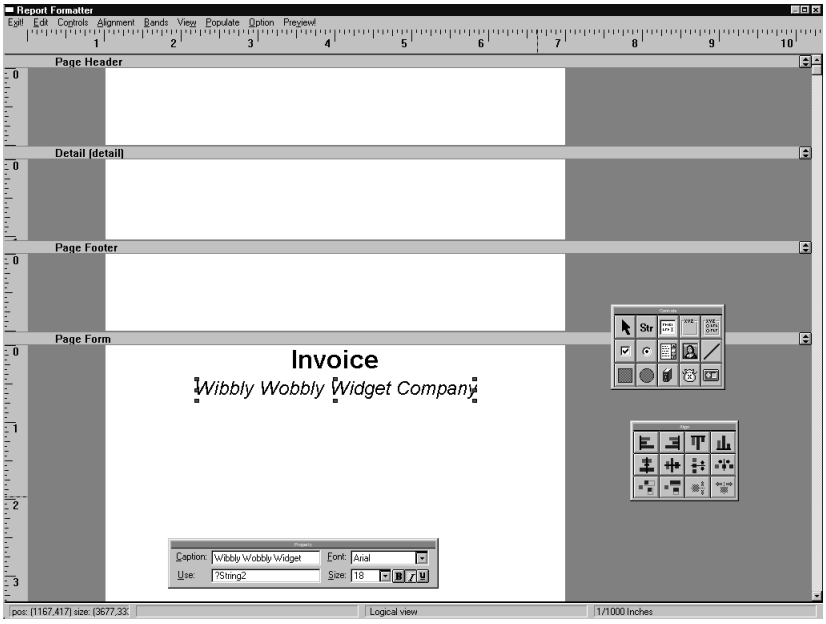
1. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
2. CLICK at the top middle of the **Page Form** band.
3. DOUBLE-CLICK on the control.

   The **String Properties** dialog appears.
4. Type *Invoice* in the **Parameter** field.
5. Press the **Font** button.
6. Select a font, style, and size to use for the text (something large and bold would be appropriate for this).
7. Press the **OK** button to close the **Select Font** dialog.
8. Press the **OK** button to close the **String Properties** dialog.
9. Resize the control so that it's large enough to hold the text, by DRAGGING its handles.

❑ Place the next string constant.

1. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
2. CLICK at the top of the **Page Form** band, just below the last string you placed.
3. RIGHT-CLICK on the control, then choose **Properties** from the popup menu.
4. Type the name of your company in the **Parameter** field.
5. Press the **Font** button and select a font, style, and size to use for the text (something just a little smaller than the previous field would be appropriate for this).

6. Press the **OK** button to close the **Select Font** dialog.
7. Press the **OK** button to close the **String Properties** dialog.
8. Resize the control so that it's large enough to hold the text, by DRAGGING its handles.



Populating the Detail Band

The Detail band will print every time new information is read from the lowest level "Child" file in the File Schematic. For this Invoice report, the lowest level "Child" file is the *Detail* file (remember that *Products* is a Many to One "lookup" file from the *Detail* file).

1. Choose **Populate ➤ Multiple Fields**, or CLICK on the *Dictionary Field* tool in the **Controls** tool box.
2. Highlight *Detail* in the **Files** list then select *DTL:Quantity* in the **Fields** list and press the **Select** button.
3. CLICK inside the **Detail** band, near its top left corner.
4. Highlight *DTL:ProdNumber* in the **Fields** list and press the **Select** button.
5. CLICK inside the **Detail** band, directly right of the first control.
6. Highlight *Products* in the **Files** list then select *PRD:ProdDesc* in the **Fields** list and press the **Select** button.
7. CLICK inside the **Detail** band, to the right of the control just placed.

8. Highlight *Detail* in the **Files** list then select *DTL:ProdAmount* in the **Fields** list and press the **Select** button.
9. CLICK inside the **Detail** band, to the right of the control just placed.
10. Highlight *LOCAL DATA InvoiceReport* in the **Files** list, then press the **New** button.

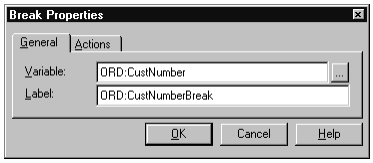
    This allows you to add a local variable without going back to the **Procedure Properties** window and pressing the **Data** button. This variable will be used to display the total price for each line item.
11. Type *ItemTotal* in the **Field Name** field.
12. Select *DECIMAL* from the **Type** dropdown list.
13. Type *7* in the **Chars** field.
14. Type *2* in the **Places** field then press the **OK** button.
15. CLICK inside the **Detail** band, to the right of the last control placed.
16. Press the **Cancel** button to exit multi-populate mode.
17. Move all the controls to the top of the **Detail** band, align them vertically, then resize the band so it is just a little taller than the controls.

Adding Group Breaks

We need to print different information on the page for each Invoice. Therefore, we need to create BREAK structures to provide the opportunity to print something every time the *Orders* file information changes and every time the *Customer* file information changes.

1. Choose **Bands ➤ Surrounding Break**, then CLICK on the **Detail** band.

   The **Break Properties** dialog appears.
2. Press the ellipsis (...) button for the **Variable** field.
3. Highlight *Orders* in the **Files** list then select *ORD:CustNumber* in the **Fields** list and press the **Select** button.
4. Type *ORD:CustNumberBreak* in the **Label** field then press the **OK** button.



A **Break (ORD:CustNumber)** band appears above the **Detail** band, which appears indented, meaning it is within the **Break** structure.

5. Choose **Bands ➤ Surrounding Break**, then CLICK on the **Detail** band.

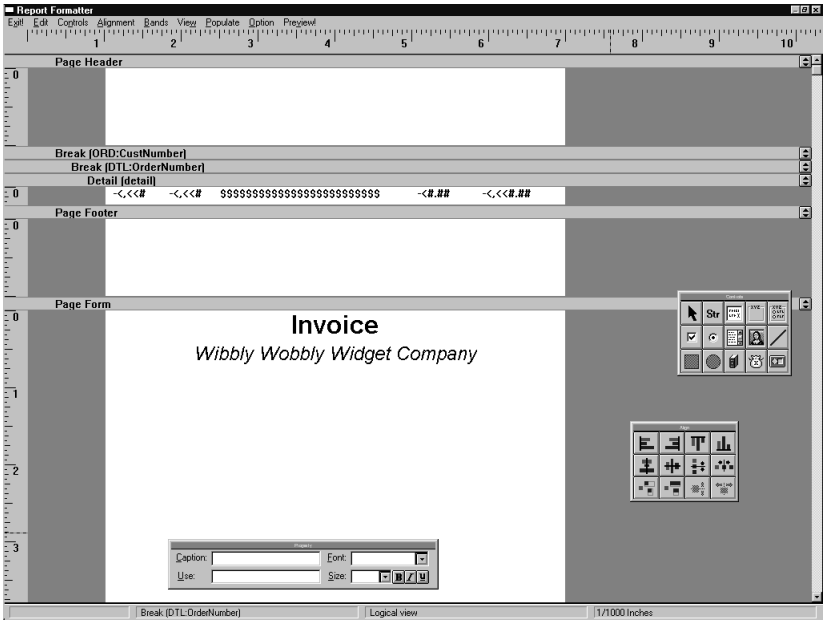
   The **Break Properties** dialog appears.

6. Press the ellipsis (...) button for the **Variable** field.

7. Highlight *Detail* in the **Files** list then select *DTL:OrderNumber* in the **Fields** list and press the **Select** button.

8. Type *DTL:OrderNumberBreak* in the **Label** field then press the **OK** button.

   Now the report design looks something like this:



❑ Create the group Headers and Footers.

1. Choose **Bands ➤ Group Header**, then CLICK on the **Break (DTL:OrderNumber)** band.

   The **Group Header (DTL:OrderNumber)** band appears above the **Detail** band. This band will print every time the value in the DTL:OrderNumber field changes, at the beginning of each new group of records. We will use this to print the company name, address, along with the invoice number and date.

2. Choose **Bands ➤ Group Footer**, then CLICK on the **Break (DTL:OrderNumber)** band.

3. RIGHT-CLICK then choose **Properties** from the popup menu.

   The **Page/Group Footer Properties** dialog appears.

4. Check the **Page after** box.

   This causes the print engine to print this band, then initiate Page Overflow. This will compose the **Page Footer** band, issue a form feed to the printer, then compose the **Page Header** band for the next page.

5. Press the **OK** button.

   The **Group Footer (DTL:OrderNumber)** band appears below the **Detail** band. This band will print every time the value in the DTL:OrderNumber field changes, at the end of each group of records. We will use this to print the invoice total.

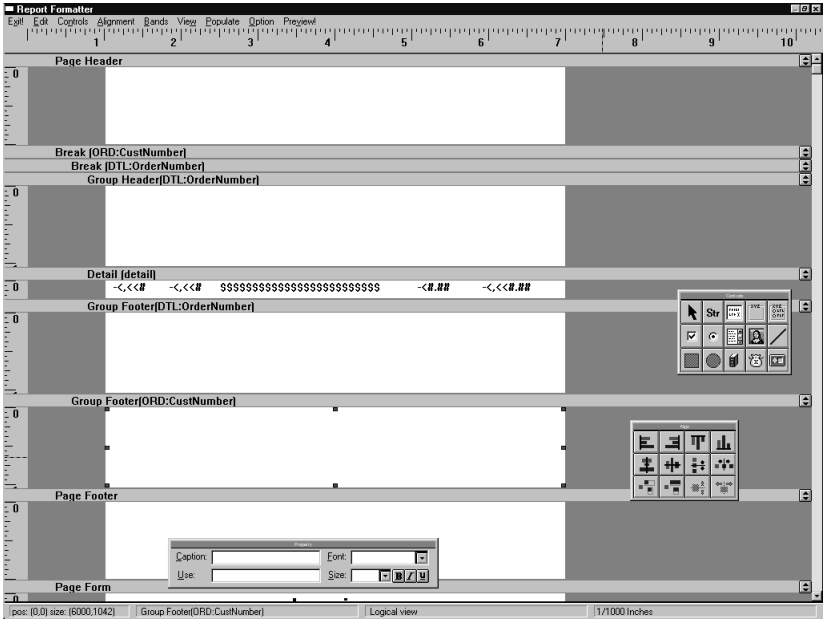
6. Choose **Bands ➤ Group Footer**, then CLICK on the **Break (ORD:CustNumber)** band.

7. RIGHT-CLICK then choose **Properties** from the popup menu.

8. Check the **Page after** box.

9. Press the **OK** button.

   The **Group Footer (ORD:CustNumber)** band appears below the **Group Footer (DTL:OrderNumber)** band. This band will print every time the value in the ORD:CustNumber field changes, at the end of each group of records. We will use this to print invoice summary information for each company.



Populating the Group Header Band

❏ Place the *Customer* file fields.

1. Choose **Populate ➤ Multiple Fields**, or CLICK on the *Dictionary Field* tool in the **Controls** tool box.
2. Highlight *Customer* in the **Files** list then select *CUS:Company* in the **Fields** list and press the **Select** button.
3. CLICK inside the **Group Header (DTL:OrderNumber)** band, near its top left corner.
4. Highlight *CUS:FirstName* in the **Fields** list and press the **Select** button.
5. CLICK inside the **Group Header (DTL:OrderNumber)** band, just below the first control.
6. Highlight *CUS:LastName* in the **Fields** list and press the **Select** button.
7. CLICK inside the **Group Header (DTL:OrderNumber)** band, to the right of the control you just placed.
8. Highlight *CUS:Address* in the **Fields** list and press the **Select** button.
9. CLICK inside the **Group Header (DTL:OrderNumber)** band, below the *second* control you placed.
10. Highlight *CUS:City* in the **Fields** list, then press the **Select** button.
11. CLICK inside the **Group Header (DTL:OrderNumber)** band, below the last control you placed.
12. Highlight *CUS:State* in the **Fields** list, then press the **Select** button.
13. CLICK inside the **Group Header (DTL:OrderNumber)** band, to the right of the previously placed control.
14. Highlight *CUS:ZipCode* in the **Fields** list, then press the **Select** button.
15. CLICK inside the **Group Header (DTL:OrderNumber)** band, to the right of the previously placed control.

❏ Place the *Orders* file fields.

1. Highlight *Orders* in the **Files** list then select *ORD:OrderNumber* in the **Fields** list and press the **Select** button.
2. CLICK inside the **Group Header (DTL:OrderNumber)** band, near its top right corner.
3. Highlight *ORD:OrderDate* in the **Fields** list, then press the **Select** button.

4. CLICK inside the **Group Header (DTL:OrderNumber)** band, below the last control you placed.

5. Press the **Cancel** button to close the **Select Field** dialog and exit multi-populate mode.

❑ Place the constant text and column headings.

1. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.

2. CLICK inside the **Group Header (DTL:OrderNumber)** band, left of the *ORD:OrderNumber* control you placed.

3. Type *Order Number:* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.

4. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.

5. CLICK inside the **Group Header (DTL:OrderNumber)** band, left of the *ORD:OrderDate* control you placed.

6. Type *Order Date:* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.

7. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.

8. CLICK inside the **Group Header (DTL:OrderNumber)** band, at the left end below the *Customer* file controls you placed.

9. Type *Quantity* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.

10. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.

11. CLICK inside the **Group Header (DTL:OrderNumber)** band, to the right of the last string you placed.

12. Type *Product* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.

10. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.

11. CLICK inside the **Group Header (DTL:OrderNumber)** band, to the right of the last string you placed, directly above the *DTL:ProdAmount* control in the **Detail** band.

12. Type *At:* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.

10. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.

11. CLICK inside the **Group Header (DTL:OrderNumber)** band, to the right of the last string you placed, directly above the *ItemTotal* control in the **Detail** band.

12. Type *Item Total* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.

❑ Place a thick line under the column headings.

1. Choose **Controls ➤ Box**, or pick the *Box* tool from the **Controls** toolbox.

2. CLICK inside the **Group Header (DTL:OrderNumber)** band, under the *Quantity* string you placed.

3. Resize the box by DRAGGING its handles until it appears to be a thick line all across the report under the column headers.

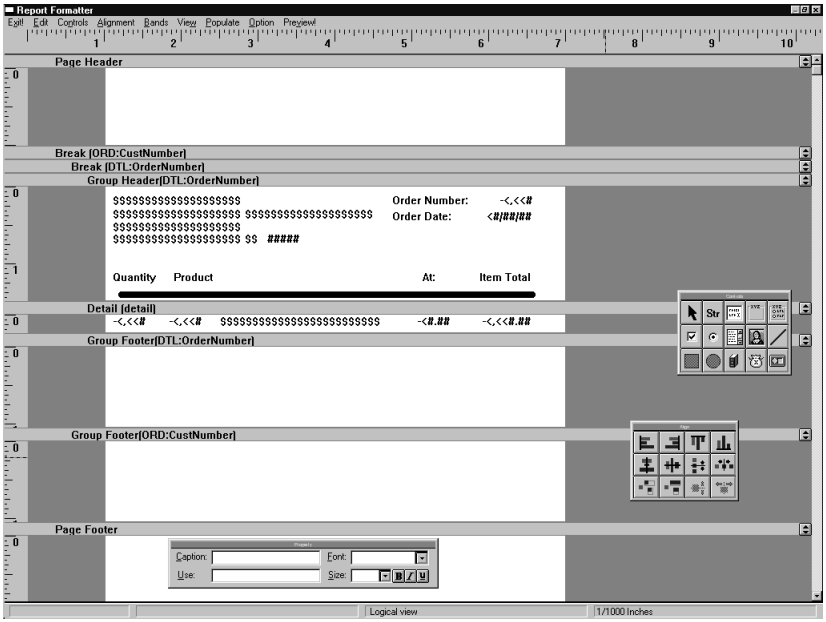
4. RIGHT-CLICK and choose **Properties** from the popup menu.

5. Select the **Extra** tab then press the **Fill Color** button.

6. Pick a color for the box from the **Fill Color** dialog (black is in the bottom left corner), then press the **OK** button.

6. Press the **OK** button to close the **Box Properties** dialog.

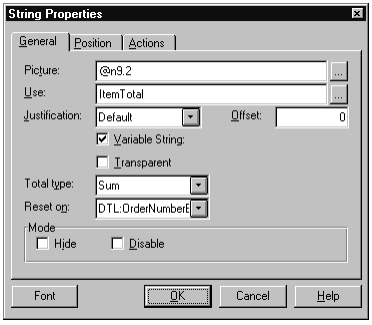
Your Report design should now look something like this:



Populating the Invoice Group Footer Band

❑ Place the constant text and total field.

1. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
2. CLICK inside the **Group Footer (DTL:OrderNumber)** band, in the middle of the band.
3. Type *Order Total:* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.
4. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
5. CLICK inside the **Group Footer (DTL:OrderNumber)** band, to the right of the string you just placed.
6. RIGHT-CLICK and choose **Properties** from the popup menu.
7. Check the **Variable String** box.
8. Press the ellipsis (...) button for the **Use** field.
9. Highlight *LOCAL DATA InvoiceReport* in the **Files** list, select *ItemTotal* from the **Fields** field, then press the **Select** button.
10. Type *@N9.2* in the **Picture** field.
11. Select *Sum* from the **Total type** dropdown list.
12. Select *DTL:OrderNumberBreak* from the **Reset on** dropdown list.



16. Press the **OK** button.

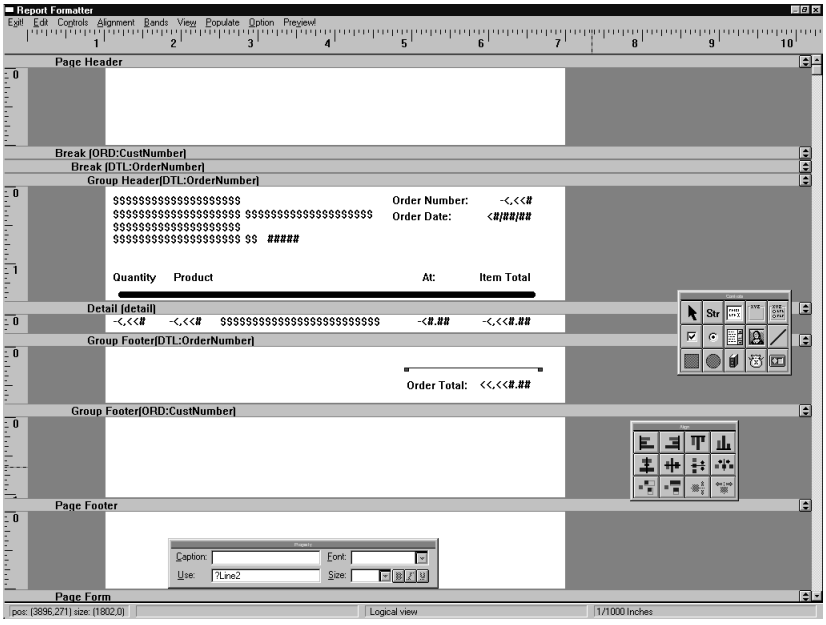
    This will add up all the *ItemTotal* contents for the Invoice and will reset to zero when the value in the *DTL:OrderNumber* field changes.

❑ Place a line above the total.

1. Choose **Controls ➤ Line**, or pick the *Line* tool from the **Controls** toolbox.

2. CLICK inside the **Group Footer (DTL:OrderNumber)** band, above the controls you just placed.
3. Resize the line by DRAGGING its handles until it appears to be above both the controls you just placed.
4. RIGHT-CLICK and choose **Properties** from the popup menu.

   If you had trouble getting the line horizontal using the mouse, just set the **Height** field to zero (0) on the control's **Position** tab.
5. Select the **Extra** tab then press the **Fill Color** button.
6. Pick a color for the box from the **Fill Color** dialog (black is in the bottom left corner), then press the **OK** button.
6. Press the **OK** button to close the **Line Properties** dialog.

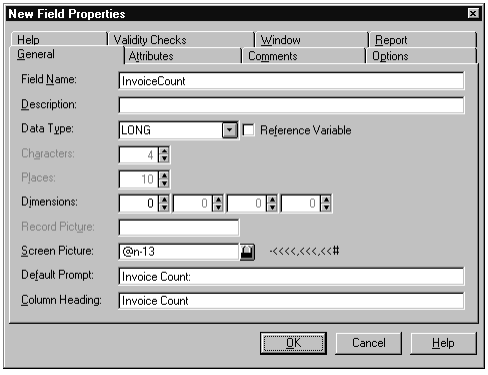


Populating the Customer Group Footer Band

1. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
2. CLICK inside the **Group Footer (ORD:CustNumber)** band, in the middle of the band.
3. Type *Invoice Summary for:* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.
4. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.

5. CLICK inside the **Group Footer (ORD:CustNumber)** band, below the string you just placed.
6. Type *Total Orders:* in the **Caption** field of the **PropertyBox** toolbox, then press TAB.
7. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
8. CLICK inside the **Group Footer (ORD:CustNumber)** band, just right of the string you just placed.
9. RIGHT-CLICK and choose **Properties** from the popup menu.
10. Check the **Variable String** box.
11. Press the ellipsis (...) button for the **Use** field.
12. Highlight *LOCAL DATA InvoiceReport* in the **Files** list, then press the **New** button.
13. Type *InvoiceCount* in the **Name** field.

    This is a field that will print the number of invoices printed for an individual company. We will write a little embedded source code to tally the count.
14. Select *LONG* from the **Type** dropdown list, then press the **OK** button.



15. Type *@N3* in the **Picture** field, then press the **OK** button.
16. Choose **Controls ➤ String**, or pick the *String* tool from the **Controls** toolbox.
17. CLICK inside the **Group Footer (ORD:CustNumber)** band, to the right of the string you just placed.
18. RIGHT-CLICK and choose **Properties** from the popup menu.
19. Check the **Variable String** box.
20. Press the ellipsis (...) button for the **Use** field.

21. Highlight *LOCAL DATA InvoiceReport* in the **Files** list, then select *ItemTotal* in the **Fields** list and press the **Select** button.

22. Select *Sum* from the **Total type** dropdown list.

23. Select *ORD:CustNumberBreak* from the **Reset on** dropdown list.

    This is the same type of total field that we placed in the *DTL:OrderNumber* group footer, but it will only reset when the *ORD:CustNumber* changes.

24. Press the OK button.

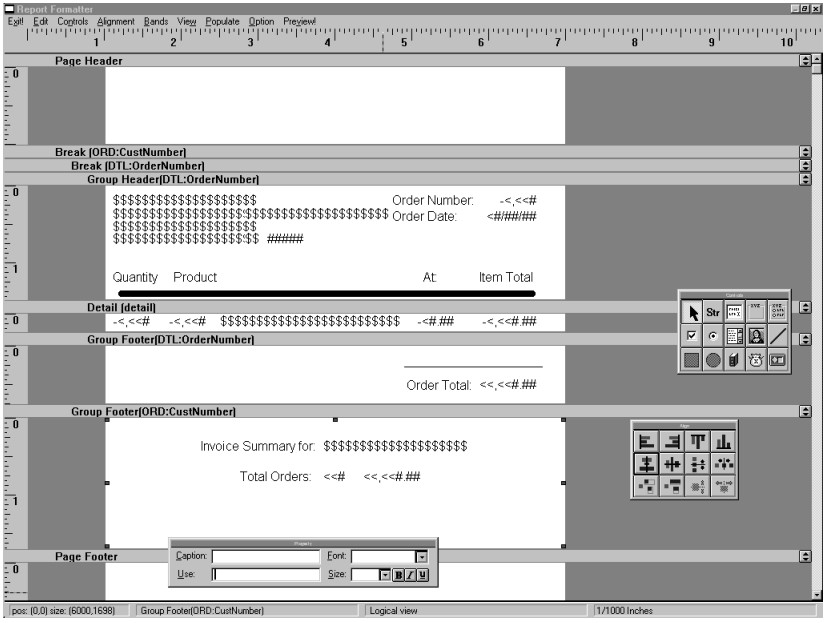
25. Choose **Populate ➤ Multiple Fields**, or CLICK on the *Dictionary Field* tool in the **Controls** tool box.

26. Highlight *Customer* in the **Files** list then select *CUS:Company* in the **Fields** list and press the **Select** button.

27. CLICK inside the **Group Footer (ORD:CustNumber)** band, just right of the *Invoice Summary for:* string you placed.

28. Press the **Cancel** button to exit multi-populate mode.

    Your report design is now complete.



28. Choose **Exit!** to return to the **Procedure Properties** dialog (be sure to save your report design).

MAKING IT WORK

To make the *ItemTotal* field contain the correct amount for each *Detail* record in the invoice, you need to add a Formula to the procedure.

❏ Add the Formula.

1. Press the **Formulas** button in the **Procedure Properties** dialog.

   The **Formula Editor** dialog appears.

2. Type *Item Total Formula* in the **Name** field.

3. Press the ellipsis (...) button in the **Class** Field.

4. Highlight *Before Print Detail* in the **Template Classes** list, then press the **OK** button.

   The *Before Print Detail* class tells the Report template to perform the calculation each time it gets ready to print a Detail.

6. Press the ellipsis (...) button in the **Result** Field.

7. Highlight *LOCAL DATA InvoiceReport* in the **Files** list, select *ItemTotal* from the **Fields** list, then press the **Select** button.

8. Press the **Data** button in the **Operands** group.

9. Highlight the *Detail* file in the **Files** list, select *DTL:Quantity* from the **Fields** list, then press the **Select** button.

   This places the *DTL:Quantity* field in the **Statement** field for you. The **Statement** field contains the expression being built, and you can also type directly into it to build the expression, if you wish.

10. Press the ***** button in the **Operators** group.

11. Press the **Data** button in the **Operands** group.

12. HIghlight the *Detail* file in the **Files** list, select *DTL:ProdAmount* from the **Fields** list, then press the **Select** button.

13. Press the **Check** button to check the expression's syntax.

14. Press the **OK** button to close the Formula Editor.

15. Press the **OK** button to close the **Formulas** dialog and return to the **Procedure Properties** dialog.

❏ Add two "save" variables.

To count the number of invoices printed for each customer, we are going to write a couple of lines of source code. This code will need two local variables to save the two Group Break fields.

1. Press the **Data** button then press the **Insert** button.
2. Type *SAV:CustNumber* in the **Name** field.
3. Select *SHORT* from the **Type** dropdown list then press the **OK** button.
4. Type *SAV:OrderNumber* in the **Name** field.
5. Select *SHORT* from the **Type** dropdown list then press the **OK** button.
6. Press the **Cancel** button to exit repetitive-add mode.
7. Press the **Close** button to close the **Local Data** dialog.

❑ Add some embed code.

1. Press the **Embeds** button.
2. Highlight **After Printing Detail Section** then press the **Insert** button.
3. Highlight *SOURCE* then press the **Select** button to call the Text Editor.
4. Type in the following code:

```
IF SAV:CustNumber <> CUS:CustNumber   !Check for customer change
  SAV:CustNumber = CUS:CustNumber     ! and reset SAV:CustNumber
  InvoiceCount = 0                    ! and the Invoice Count
END
IF SAV:OrderNumber <> ORD:OrderNumber !Check for Order change
  SAV:OrderNumber = ORD:OrderNumber   ! and reset SAV:OrderNumber
  InvoiceCount += 1                   ! and increment Invoice Count
END
```

   This code checks for a change in the Customer number. When it detects that it has changed it saves the new Customer number and resets the *InvoiceCount* to zero. It then checks for a change in the Order number. When it detects that it has changed it saves the new Order number and increments the *InvoiceCount* by one.

5. Choose **Exit!** to return to the **Embedded Source** dialog.
6. Press the **Close** button to return to the **Procedure Properties** dialog.
7. Press the **OK** button in the **Procedure Properties** dialog to close it.
8. Choose **File ➤ Save**, or press the *Save* button on the toolbar to save your work.

What's Next?

Congratulations, you made it to the end of the tutorial! Welcome to the growing community of Clarion developers!

While this tutorial application is by no means a "shrink-wrap" program, it has served to demonstrate the "normal" process of using Clarion's Application Generator and all its associated tools to create a Windows application that actually performs some reasonably sophisticated tasks. Along the way, you have used most of the high-level tool set, and seen just how much work can get done *for you* without writing source code. You have also seen how just a *little* embedded source can add extra functionality to the template-generated code.

So where should you go from here to learn more?

The best place to go next (besides creating an application of your own design) are the chapters of the *User's Guide* that describe the standard Clarion template set: *Using the Procedure Templates* and *Using the Control, Code, and Extension Templates*. These two chapters explain in detail all the capabilities of the templates available to you in the Application Generator. There are also many example programs on your disk that demonstrate some of the "tricks" you can use in your Clarion programming.

We also suggest you look at the generated code for this tutorial program using the Text Editor, to see the Clarion language code the Application Generator "wrote" for you. You'll find it in the TUTOR*.CLW files in the \CW15\TUTORIAL directory. The *Language Reference* is your "bible" for any questions on individual code statements, and it is also your gateway to learning the more low-level capabilites of Clarion for Windows.

We also highly recommend joining the on-line community of developers who regularly visit our forum on CompuServe (GO TOPSPEED). The level of advice and assistance that these developers (and TopSpeed's Technical Support staff) provide to each other is unmatched in the industry! The forum always has the latest information on new update releases and upcoming products, along with invaluable help from other developers who have already "been there, done that," which makes it well worth the effort to connect.