ReportWriter

SoftVelocity

SoftVelocity Incorporated
2769 East Atlantic Blvd.
Pompano Beach, Florida 33062
(954) 785-4555
www.softvelocity.com

Printed in the United States of America (0703)

Contents

1 - GETTING STARTED

Welcome to the Fast Track for Report Development

Welcome to Clarion ReportWriter. You have just purchased what SoftVelocity Incorporated believes is *one of the most powerful reporting tools on the market*! You can now create sophisticated reports from your data files faster than you ever thought possible. This revolutionary development environment will dramatically increase your productivity. You can use practically any existing database effortlessly.

Does this sound too good to be true? Just do the Report Wizard Tutorial in chapter four of this book, and we're sure you'll be convinced in less than an hour.

Clarion ReportWriter is a very complete and comprehensive product; however, you can master it one step at a time. Before you know it, you'll be creating reports "at the speed of light."

ReportWriter's unique features enable you to:

- Import a Clarion Database Dictionary containing multiple related data files, even if they have different file formats. ReportWriter can read TopSpeed, Clarion, Btrieve, Scalable, dBase III or IV, FoxPro, Clipper, ODBC, ASCII, BASIC, and DOS files.
- Sort data any way you choose and retrieve only those records you select.
- Import your existing data files' definitions, from most supported file formats and link them using User-defined relationships.
- Perform mathematical calculations on your data.
- Create "error-proof" formulas, conditions, and computed fields with the Formula Editor.
- Create "runtime" entry fields—values you enter just before you run a report. These values can even be used to perform computations or filters on your data.
- Use memo fields and image fields.
- Protect your reports and databases with passwords.
- Modify your reports to print dates, text, and numbers in any format.
- Organize any type of information. You can make a wide variety of reports including: inventory reports, payable and receivable reports, customer, client, and vendor reports, mailing list labels, form letters, personnel reports... The list goes on and on.
- Easily print on labels or pre-printed forms.
- Control your report's pagination easily, for example, printing a section of your report on its own page.
- Design reports in a WYSIWYG (What You See, Is What You Get) formatter. All of your report's elements are shown right on the screen.
- Create multiple detail bands. You can print a different detail band for different situations, in the same report.

- Run a report directly from a Windows Icon or shortcut. You can even supply passwords and runtime field values through an Initialization File (.INI).

- Immediate Results—Build *Your* Reports in minutes.

  Clarion ReportWriter for Windows' Report Wizards will build a report from a Clarion dictionary, a data file, or a database. Specify a few options, and your report is ready for the printer.

- Database Independence

  New high performance database drivers support popular databases and accounting packages; including Xbase formats, 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 supports as well).

  Use 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

This manual is divided into the following parts:

- Getting Started

  Chapters 1 and 2 provide an overview of ReportWriter and instructions for installation.

- Tutorials

  Chapters 4 through 6 are step-by-step tutorials.

- Reference

  Chapters 7 through 9 contain step-by-step instructions for every task you can perform in Clarion ReportWriter. Each chapter contains a full explanation of the steps to perform a task. This is a reference section.

- Advanced Topics

  Chapter 10 provides instructions to use the advanced features in ReportWriter, including the command line parameters for the print engine and instructions for deploying your reports to other users.

  Chapter 11 provides instructions for Clarion Programmers who want to extend ReportWriter's functionality or integrate ReportWriter into Clarion applications.

- Appendixes

  These are references for specific features of Clarion ReportWriter.

- Glossary and Index

  The Glossary contains explanations of many of the terms used throughout this manual. The index lists most of the topics covered in this manual.

Documentation Conventions

The documentation uses the typeface and keyboard conventions that appear below.

Typeface Conventions

Italics Indicates what to type at the keyboard or select from a list, 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.

`Courier New` 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 f1key.

ALT+X Indicates a combination of keystrokes. Hold down the alt key and press the x key. Then release both keys.

Where to Find More Information

In addition to this book, there is an extensive on-line help system available for Clarion ReportWriter.

Important: if any part of the on-line help text conflicts with the printed documentation, the on screen help takes precedence. SoftVelocity Incorporated 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.

Technical Support

Help can be obtained from several different online newsgroups. Our web site, www.softvelocity.com, details the available technical support plans.

Usenet Newsgroup--comp.lang.clarion

You can participate in the Clarion Usenet Newsgroup on the Internet--comp.lang.clarion. In this newsgroup, Clarion programmers from around the world exchange ideas and techniques. Log into your News Server and subscribe to comp.lang.clarion. If your news server does not carry the feed, you should contact your Internet provider.

SoftVelocity's product newsgroups

SoftVelocity's internal news server offers newsgroups for all SoftVelocity products. To subscribe to these groups use news.softvelocity.com as the news server. There are several newsgroups you can subscribe to on this server.

SoftVelocity's Web Site:

You can find other Clarion resources on the Internet by visiting SoftVelocity's site on the World Wide Web:

http://www.softvelocity.com

Paid Technical Support

Paid telephone technical support is available. Refer to the SoftVelocity web site for the most up to date information on the available technical support plans.

Product Information

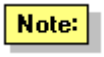
Registering This Product

Before you begin using Clarion ReportWriter, 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.

Distribution Policy

Clarion ReportWriter Report Library files (.TXR) are redistributable along with any version of the print engine (C60PRNTx.EXE, C60PRLIB.DLL, or C60PRLBX.DLL) and the supporting DLLs. This allows you to distribute the files needed for end users to print or preview reports. It does not allow any modifications to reports.

To design reports, each user must own a separate copy of Clarion ReportWriter. You can purchase ReportWriter from TopSpeed for resale to your clients or to include with your application and Report Libraries as part of a complete solution. This allows them to modify your reports or create new ones.



ReportWriter (C60RW.EXE) is not redistributable. Each user must purchase a separate copy.

See *Distributing your Reports* in Chapter 10—*Advanced Topics* for details on deploying Report Libraries and the necessary supporting files.

2 - SETUP

System Requirements

You can run Clarion ReportWriter on any system that meets the minimum system requirements for Microsoft Windows 95™, Windows 98™, Windows NT 3.51 (or higher), Windows 2000, and Windows NT.

- Windows 95/98, 12 Megabytes of RAM recommended.
- Windows NT, 16 Megabytes of RAM recommended.
- Minimum of 8 Megabytes free hard disk space, depending on the Setup options you select.

The reports you create with ReportWriter will execute using the distributable print engine (C60PRNTX.EXE) on computers that meet only the minimum requirements for these operating systems.

The Setup Program

The Setup program, on the CD-ROM of your installation set, decompresses and copies the ReportWriter 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.
- It asks before updating the PATH statement in your AUTOEXEC.BAT file to include the ReportWriter directory.
- In Windows 95/98 or NT, it creates Shortcuts on the Start Menu for the ReportWriter environment and Help file.

Starting Setup

To start the ReportWriter Setup program in Windows 95/98:

1. Insert the CD-ROM in its drive.
2. From the Start menu, choose **Settings** ▸ **Control Panel**.
3. Choose **Add/Remove Programs** then press the **Install** button.

   The Windows 95/98 Wizard directs 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 for the portions you want to install, then press the **OK** button.
2. Specify the target drive and directory, then press the **OK** button.

   Setup installs the main components of ReportWriter to subdirectories below the target directory you specify in the dialog.

   The ReportWriter 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 display as Setup copies the files.
3. Choose *Yes* or *No* when Setup asks whether to modify the PATH for you.

   The ReportWriter 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 ReportWriter 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 help file.

   If you don't wish to read it right away, you'll find an icon for Late Breaking News in ReportWriter's on-line help. We recommend reading it as soon as Setup has copied all the files.
5. Press the **OK** button when Setup is done.

Starting Clarion ReportWriter

Under Windows 95/98, choose Start ▸ Programs ▸ Clarion ▸ ReportWriter.

The Clarion ReportWriter 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 Chapters 7 and 8.
- Go on now with the rest of this book. In chapter 4, you'll create three impressive reports—from data files, to your printer—all in *minutes*.

3 - DATA BASICS

Anatomy of a Database

A database is a collection of information (data) in a system of files, records, and fields. The database is maintained by one or more computer programs or applications.

The basic unit of data storage is a **field**. A field is a storage place for information of a similar type. For example, one field might store a last name and another field might hold a telephone number.

A group of different fields that are logically related make up a **record**. A record contains all the information related to one subject. For example, all the fields containing information concerning one student (name, address, telephone number, student number, etc.) makes up one student's record. This would be similar to a file folder a school might keep for each student.

A collection of logically related records make up a **file**. Using the same example, a collection of all students' records makes up the student body file. This would be similar to the file cabinets where students' folders are kept.

Another way of looking at this is as a table or spreadsheet:

Number	First Name	Last Name	Major
206-65-7223	Dan	Headman	English
168-91-7542	Gary	Pack	English
105-22-0863	Diane	Quinn	English
141-02-7461	Alvin L	Bailey	Computer Science
123-44-9999	Deb	Brown	Computer Science
101-71-0530	Jerry	Cade	Computer Science
180-43-9184	Nina	Robb	Sociology
229-87-5138	Steve	Stone	Sociology
188-67-7396	Robin	Bailey	Business
164-10-8264	John	Uber	Business
207-95-3939	Brian	Wilcox	Business
116-62-4660	Robert	Macdonald	Law

In this format, the entire table is a file, each row is a record, and columns represent fields.

A database is a collection of related files (tables). This is similar to a bank of file cabinets where the entire school records are kept. One file cabinet might hold the files with students' personal data, another with class enrollment information, and another with faculty information.

A **database** is a collection of tables with defined relationships between them. Effective database design breaks the data into related files that are joined together through linking fields. This will be covered in detail later in this section.

Summary:

One or more fields combine to form a record.

One or more records combine to form a file.

A collection of related files is a database.

File Systems and File Drivers

There are several data file formats used on PCs. These are the actual physical storage formats written to disk by programs that maintain the data files. Through the use of TopSpeed's file driver technology, Clarion ReportWriter supports many of them. File drivers enable ReportWriter to read these different file formats. ReportWriter also allows file drivers to be added as they are needed.

The file drivers provided with ReportWriter, include: TopSpeed, Clarion, ODBC, Btrieve, Clipper, dBase III & IV, FoxPro, ASCII, BASIC, and DOS. When you need to read data from another file system, you can add new file drivers. Call SoftVelocity's Sales department at (877) 733-4555 or (954)785-4555 to inquire about the availability of any specific file driver you need.

Each file system has its own idiosyncrasies and limitations. See the Database Drivers Appendix for more information.

Data Types

Fields can store many different types of data, but each individual field may hold only one type. When a field is defined, its data type is specified. This enables it to efficiently store that type of data. For example, to store a number from 0 to 100, using a field defined as a single BYTE takes less space than one defined as a decimal number field (a byte can hold an unsigned whole number between 0 and 255).

ReportWriter supports the following data types:

Alphanumeric

STRING A field designed to hold a specific number of alphanumeric and other ASCII characters.

PSTRING A field designed to hold a character string, with a leading length byte included in the number of bytes declared for the string. This is the type used by the Pascal language and the "LSTRING" data type of the Btrieve Record Manager.

CSTRING A field designed to hold a character string terminated by a null character—ASCII zero (0). This is the type used in the "C" language and the "ZSTRING" data type of the Btrieve Record Manager.

Integers (whole numbers)

BYTE A field designed to hold an unsigned (positive) integer from 0 to 255.

SHORT A field designed to hold a value from -32,768 to 32,767.

USHORT A variation of the SHORT field data type. USHORT is a two-byte field designed to hold a value from 0 to 65,535. Negative numbers are not allowed in a USHORT field.

LONG A field designed to hold a value from

-2,147,483,648 to 2,147,483,647.

ULONG A variation of the LONG field data type. ULONG is a four-byte field designed to hold a value from 0 to 4,294,967,295. Negative numbers are not allowed in a ULONG field.

Floating Point (Real) Numbers (numbers with fractional portion)

REAL A field designed to hold an eight-byte signed floating-point number with 15 significant digits.

SREAL A field designed to hold a four-byte signed floating-point number with 6 significant digits. An SREAL field uses the Intel 8087 short real (single precision) format.

BFLOAT8 A variation on the REAL data type, a BFLOAT8 field is designed to hold an eight-byte signed floating point number using the Microsoft BASIC (double precision) format.

BFLOAT4 A variation on the REAL data type, a BFLOAT4 field is designed to hold a four-byte signed floating point number using the Microsoft BASIC (single precision) format.

Packed Decimal Numbers

DECIMAL A field designed to hold a signed packed decimal format value from

-9,999,999,999,999,999,999,999,999,999,999 to 9,999,999,999,999,999,999,999,999,999,999.

PDECIMAL A field designed to hold a signed decimal number in the Btrieve and IBM/EBCDIC packed decimal type of format. The only difference between a DECIMAL and a PDECIMAL is the place it stores the sign.

Other Data Types

DATE A four-byte field designed to hold a date variable in Btrieve Record Manager format.

TIME A four-byte field designed to hold a time variable in Btrieve Record Manager format.

GROUP A field made up of two or more fields. For example, a GROUP field called PhoneNumber could be made up of two fields: AreaCode and Phone. A Group Field gives you the option of using the entire group or any of its components.

PICTURE Although PICTURE is not a data type, when selected, PICTURE declares a STRING data type with a picture token (such as @P###P) defining the number of characters in the string. The picture token used for the field declaration is entered in the Record Picture field. This is useful for data fields whose storage picture and display picture must be different.

Sorting Data: Keys and Indexes

One of the most powerful aspects of a computerized database is the ability to sort data in many different ways. To do this manually requires multiple copies of record forms, many file folders, and many file cabinets. It would also require a lot of time spent filing each copy in different places for each sort order.

Sorting computer records in a database merely requires the definition of keys or indexes. Keys and indexes declare sort orders other than the physical order of the records within the data file. In some file systems the keys are kept in separate disk files, in others they are contained within the same file as the data. TopSpeed's file driver technology handles these differences transparently.

Keys and indexes are functionally equivalent. The only difference is the way they are maintained by an application.

- A **key** is dynamically maintained by the application. Every time a record is added, modified, or deleted, the sort sequence is updated, if necessary.
- An **index** is a sort sequence that reflects the contents of a data file at a particular time. An index is not dynamically maintained, it is only built when specified. Indexes are useful to create sort sequences that are infrequently used, such as month-end or year-end processes.

A dynamic index is a sort sequence created at runtime. The user-defined sorts in ReportWriter are dynamic indexes. They are defined for a specific report and are automatically built each time that report runs.

Using the student file example discussed earlier, suppose you wanted to sort the students' records two ways: by name alphabetically, and by name within each major. This produces two alternate sorts.

The first example uses a one-component key: the student name.

Sorted Alphabetically by Name

Number	First Name	Last Name	Major
141-02-7461	Alvin L	Bailey	Computer Science
188-67-7396	Robin	Bailey	Business
123-44-9999	Deb	Brown	Computer Science
101-71-0630	Jerry	Cade	Computer Science
206-65-7223	Dan	Headman	English
116-62-4660	Robert	Macdonald	Law
168-91-7542	Gary	Pack	English
105-22-0863	Diane	Quinn	English
180-43-9184	Nina	Robb	Sociology
229-87-5138	Steve	Stone	Sociology
164-10-8264	John	Uber	Business
207-95-3939	Brian	Wilcox	Business

The second example uses two components in the key: major and student name. **A key can contain one or more fields, allowing sorts within sorts.**

Sorted by major, alphabetically within major

Number	First Name	Last Name	Major
206-65-7223	Dan	Headman	English
168-91-7542	Gary	Pack	English
105-22-0863	Diane	Quinn	English
141-02-7461	Alvin L	Bailey	Computer Science
123-44-9999	Deb	Brown	Computer Science
101-71-0630	Jerry	Cade	Computer Science
180-43-9184	Nina	Robb	Sociology
229-87-5138	Steve	Stone	Sociology
188-67-7396	Robin	Bailey	Business
164-10-8264	John	Uber	Business
207-95-3939	Brian	Wilcox	Business
116-62-4660	Robert	Macdonald	Law

Ascending and Descending Sort Orders

Some file systems support both ascending and descending order for keys or components of keys. Other file systems only support ascending order, which means the data can only be sorted from lowest to highest.

ReportWriter allows you to create a user-defined sort sequence, in either ascending or descending order, no matter what file system you are using. This feature overcomes the limitation of some file systems' ascending-only keys. Descending order keys can be useful. For example, if the records are sorted on a date field, you might want to see the most recent record first.

ReportWriter also gives you the ability to "mix and match" ascending and descending sort orders in user-defined sorts. For example, you could sort the student records by descending graduation year (the most recent first), and alphabetically (ascending) within those years.

By Graduation Year (descending), alphabetically within year

Grad Year	Last Name	First Name
1999	Babbitt	Jim
1999	Headman	Dan
1998	Bailey	Alvin L
1998	Bailey	Robin
1998	Brown	Deb
1998	Quinn	Diane
1998	Stone	Steve
1998	Wilcox	Brian
1997	Cade	Jerry
1997	Macdonald	Robert
1997	Pack	Gary
1997	Robb	Nina
1997	Uber	John

Using Keys as Range Limits

Suppose you want to create a class enrollment report from a database containing records for the past fifteen years. All you are interested in (for purposes of this report) is the last three years. You can dramatically reduce processing time if you use a subset of the data file that only contains the records from the last three years.

Two steps are needed to accomplish this:

- First, define a key to sort the data by the date of each course.
- Next, define the range limits you are interested in. A range limit specifies a subset of the entire file to process. Only those records that fall within the range limits are considered.

In this example, only one-fifth of the records are processed (assuming that each year's course offerings are the same). Reducing the number of records to consider by 80% reduces processing time by the same amount.

ReportWriter also lets you create multiple ranges. This allows you to specify a subset of the records, with more than one range of records in the subset. For example, the student records might be sorted by the student's home state. If you wanted to consider just the records of students from New York and New Jersey, those two ranges could be specified.

This could also be accomplished with a record filter, but using a subset of the data file reduces processing time.

Relationships Between Files

One goal of relational database design is reducing data redundancy. The basic rule is that data should be located in only one place. This is beneficial in two ways. First, it reduces storage space requirements. Second, it makes the database easier to maintain. To reach this goal, data files are broken up into separate, related files through a process called data normalization.

The first step is to move any repeating groups into separate files. For example, if a student could take a maximum of six classes, you could design the student file to contain six class fields (for class1, class2, class3, etc.). But, not all of these fields would be used in each record. If one student is taking six courses, all would be used, but if another student only took one class, there would be five empty fields in his record. For that reason, the class fields are moved into a separate file, eliminating the need to reserve space for empty fields.

The next step is to move any redundant data into separate files. Every field in the student file must be dependant on the primary key (Student Number). The student's name, address, and phone number would remain in the student file. But the student's major's data would be moved to a separate file. This eliminates the need to repeat the Major's Description Field for each student that has that Major. To accomplish this, a Major number field is added to the student file and is used to create a link to the Majors file.

Once data storage is "normalized," related information is linked by ensuring a field in one file is identical to a field in the other. These common fields create the relations between files. A linking field could be a student number, a course code, or a classroom number. Any field (or group of fields) that uniquely identifies a record in the primary file can be used as a link.

Examples of these relations can be found in the example of the school database:

- One teacher teaches many classes (One-to-Many).
- Many students would have one major (Many-to-One).

Summary

- A database is a collection of information (data) in a system of fields, records, and files.
- Each data item should be stored in only one place.
- One or more fields makes up a record. One or more records make up a file. A collection of related files make up a database.
- ReportWriter can access many different file systems through the use of file drivers.
- Although not all file drivers support both ascending and descending sort orders, ReportWriter lets you create either type in user defined sorts.
- Keys and indexes declare sort orders other than the physical order of the records within the data file.
- A key can contain more than one field, allowing sorts within sorts.
- Files are joined together in relationships through common fields containing identical data.
- Fields can store many different types of data, but each individual field is specified to hold only one type.
- Range Limits enable you to process a subset of records, which reduces processing time.

4 - TUTORIAL: CREATING SIMPLE REPORTS

Guide to Tutorial

In this tutorial, you will create some simple reports from existing data files. The files are in the \Clarion6\RWTUTOR subdirectory (unless you specified another base installation directory during installation). The completed tutorials are in the \Clarion6\RWTUTOR\SOLUTION subdirectory.

Objectives

After completing this lesson, you will know how to:

- Start ReportWriter
- Create a Report Library based on a Data File
- Create reports
- Design the report's layout
- Place fields
- Format data using Computed Fields
- Place page headers on a report
- Use horizontal and vertical lines to separate elements of a report
- Preview a report
- Print a report
- Save a report
- Copy and modify reports
- Change a report's sort order
- Move and Delete controls
- Use the Label Wizard to create Mailing Labels

Creating and Editing Reports

Starting ReportWriter

After installing ReportWriter:

1. Choose **Start** ▸ **Clarion** ▸ **ReportWriter**.

Creating a New Report Library File

1. Press the  button on the toolbar, (or choose **File** ▸ **New**).

   The **New Report Library** window appears. This is where you specify the Report Library's properties.

2. In the **Filename** field, type *RWTUTOR*.

   ReportWriter automatically adds the .TXR extension for the file.

3. In the **Working Directory** entry, type *C:\Clarion6\EXAMPLES\RWTUTOR* (the path of the working directory) or press the ellipsis (...) button to select it from a standard file dialog.

   If you specified a different directory during installation, modify the entry as needed. The working directory is the first directory in which ReportWriter looks for your data files.

4. In the **Description** entry box, type a description for this Report Library.

   Although optional, the description can help you identify the Report Library.

5. Skip the **Password** entry.

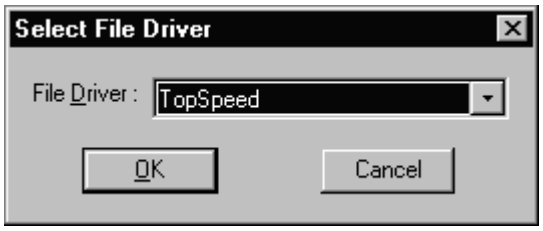
   You can use password protection for a Report Library by specifying a password here. For this tutorial, do not use password protection.

6. In the **Create Using** section, mark the **Database** radio button.

   This specifies that the Report Library is based on a data file instead of a Data Dictionary or Report Library. Using a Data Dictionary as the basis of a Report Library is preferable because more information is stored in a Data Dictionary than individual files (e.g., relationships between files). However, you can create the same reports on data files without a Data Dictionary. This is discussed further in *Chapter 7—Using Report Libraries*.

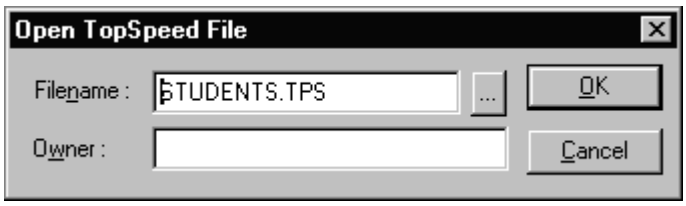
7. Press the ellipsis (…) button next to the **Filename** entry box.

   The *Select File Driver* dialog appears.



8. Select *TOPSPEED* from the **File Driver** drop-down list, then press the **OK** button.

   The *Open TopSpeed File* window appears.



9. Press the ellipsis (…) button next to the **Filename** entry box.

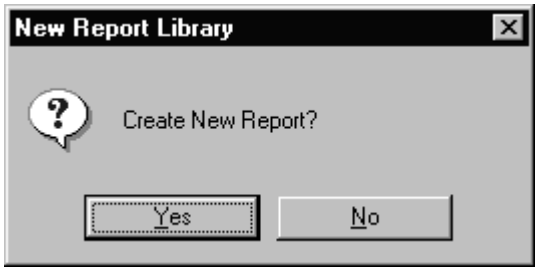
   A standard file open dialog appears.

10. Select *STUDENTS.TPS*, then press the **OK** button (if necessary, "walk" the directory tree to the *\CLARION6\EXAMPLES\RWTUTOR* subdirectory).

11. Press the **OK** button on the *Open TopSpeed File* window.

12. Press the **Create** button to make the Report Library.

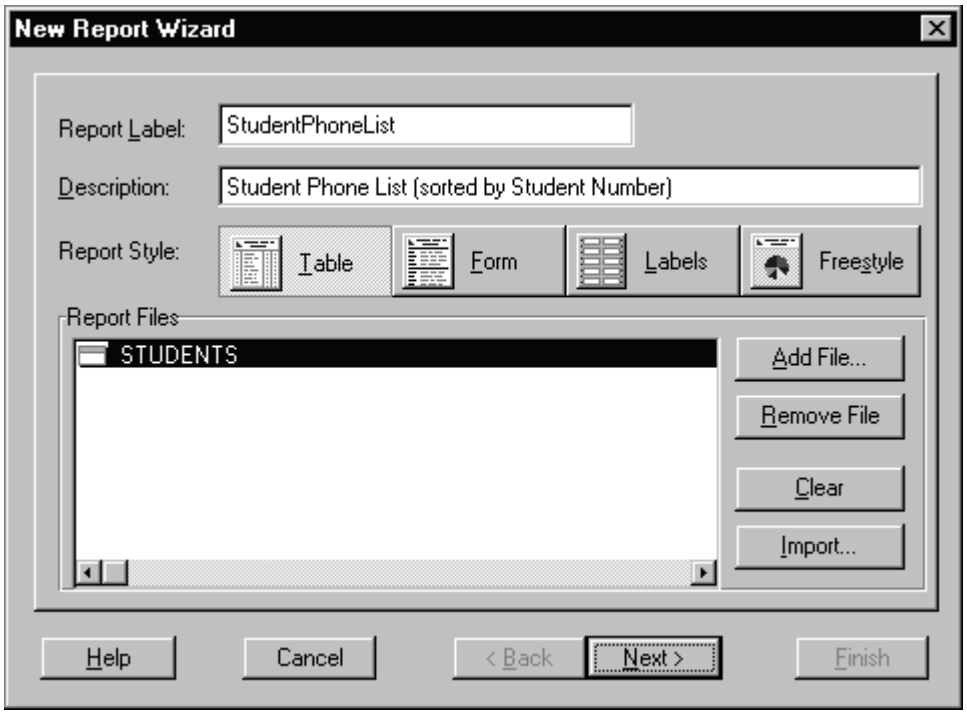
    Since this Report Library does not yet contain a report, ReportWriter asks if you want to create one now.



13. Press the **Yes** button to create a report.

Creating a New Report—Using the Report Wizard

The **Report Wizard** appears. This wizard creates the report for you, basing the report layout on what you specify on its screens.



1. In the **Report Label** entry, provide a unique label (name) for the report, as shown above.

   Each report must have a unique label. This identifies the report to the print engine when previewing or printing a report. Labels must begin with a letter and can contain any combination of letters, numbers, the underscore character (_), or the colon character (:). Blank spaces are not allowed.

2. In the **Description** entry, type a description for the report.

   Although this is not required, a description helps you to identify the report. This description appears in the Report Library's list of reports.

3. Choose the *Table* **Report Style** by selecting the 

button.

   You will use the other styles in later parts of the tutorial.

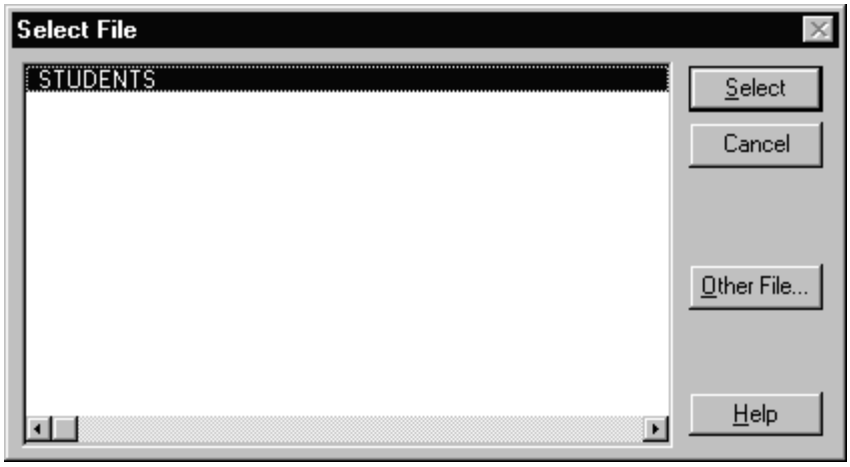
4. Specify the Files the report uses (go on to *Specifying the File Schematic*).

Specifying the File Schematic

The File Schematic specifies the file(s) used for the report. Each report has its own File Schematic. Reports can use one or more files. Before specifying a schematic, decide which file(s) you will be using for the report. In this part of the tutorial, you will use only one file—the *STUDENTS* file.

1. Press the **Add File** button.

   The *Select File* window appears. Since the STUDENTS file is the only file in the Report Library it is the only one listed.



2. Highlight *STUDENTS*, then press the **Select** button.

   This places the Students file in the file schematic.

3. Press the **Next>** button.

   The *Fields to Populate* window appears.

Populating Fields—Using the Report Wizard

In this dialog, you specify the fields to place on the report. You select fields by adding them to the Selected Fields list.

1. Highlight *Number*, then press the  button.

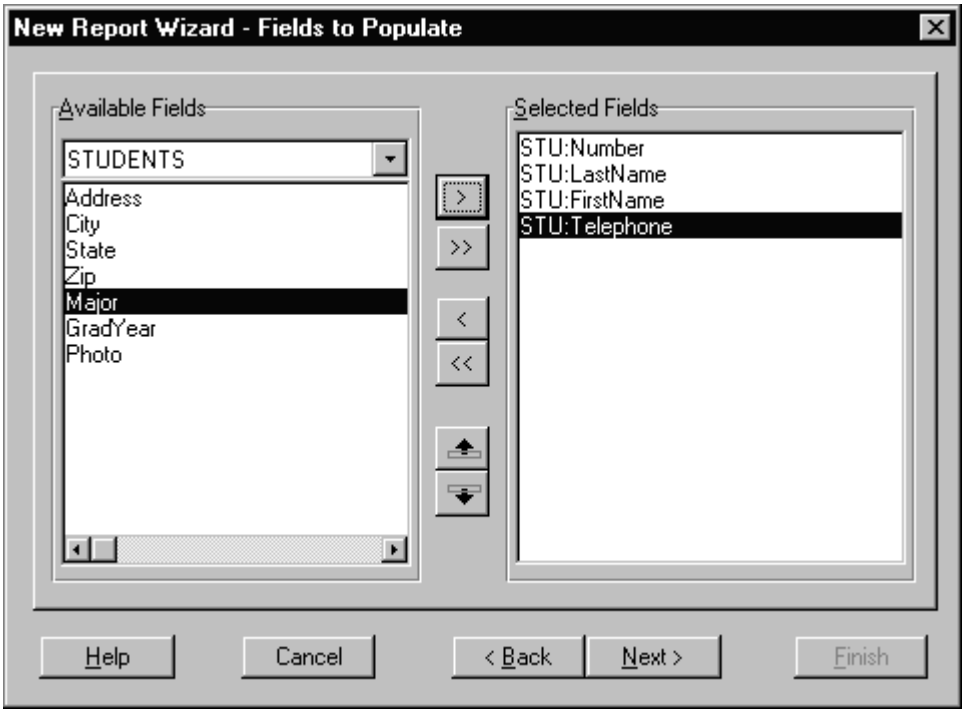
   The *STU:Number* field appears in the **Selected Fields** list.

2. Highlight *LastName*, then press the  button.

   The *STU:LastName* field also appears in the **Selected Fields** list.

3. Highlight *FirstName*, then press the  button.

4. Highlight *Telephone*, then press the > button.



5. Press the **Next>** button.

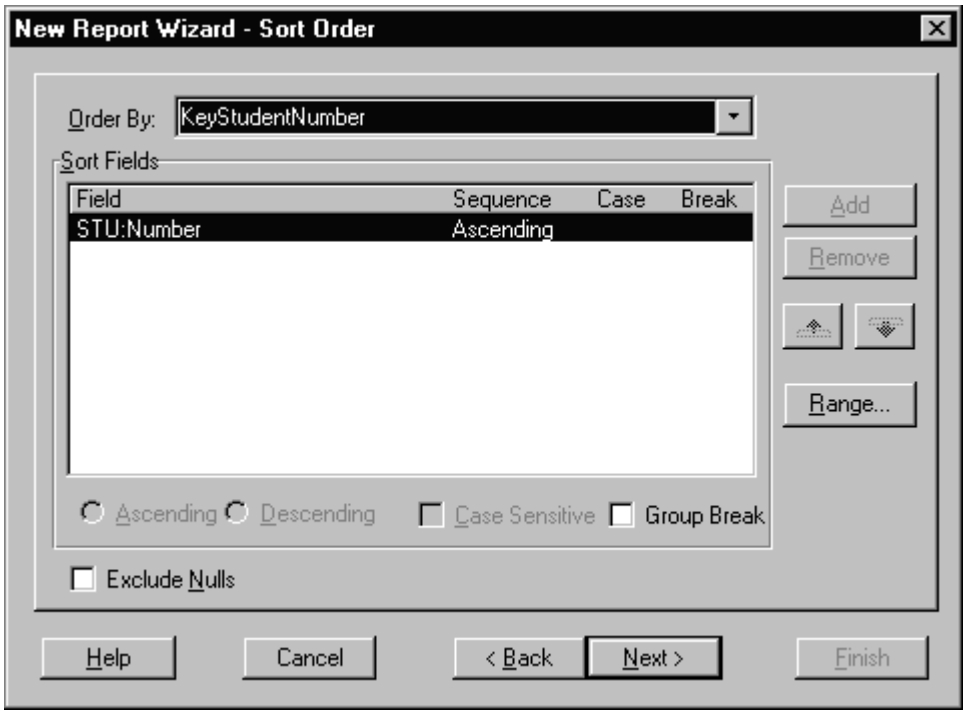
The Sort Order window appears.

Specifying the Sort Order—Using the Report Wizard

In this window, you specify the sort order and group breaks for a report. You can sort your report using an existing Key or define a new user-defined sort order.

1. Use the **Order By** drop-down list to select KeyStudentNumber.

This key sorts the report by students' Student Number.



2. Make sure the **Group Break** box is not checked.

This check box lets you group records together. You'll use this later in the tutorial.

3. Press the **Next>** button.

The *Report Layout* window appears.

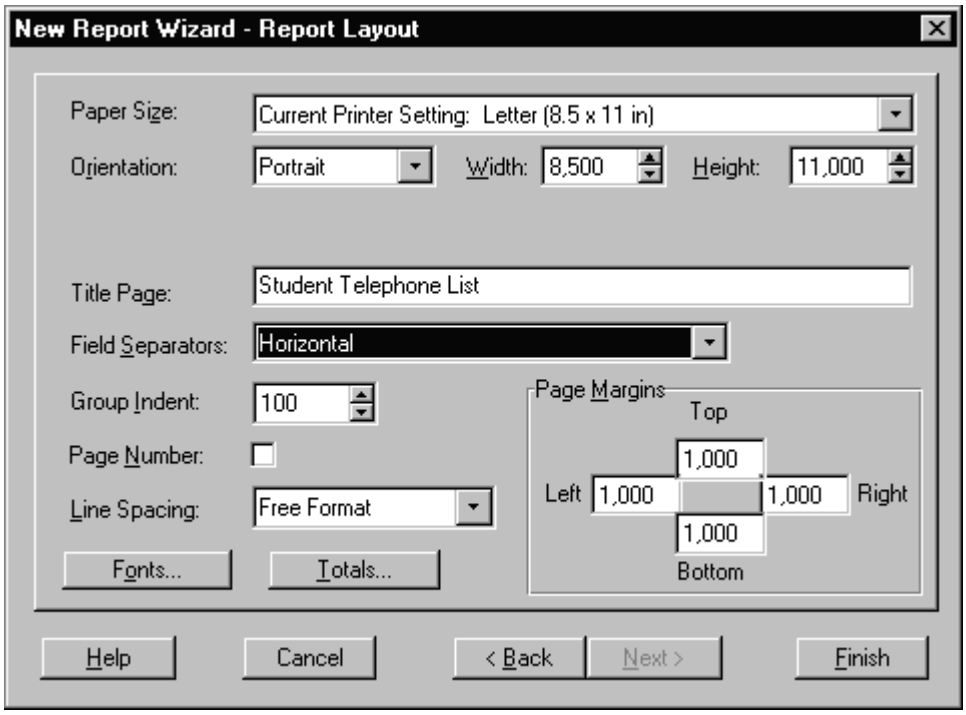
Specifying the Report Layout—Using the Report Wizard

This window lets you specify a number of options to fine-tune your report's layout. For this tutorial, you will use most of the default settings.

1. Skip the **Paper Size** field.

This field allows you to specify one of a number of common paper sizes. In most cases (and for this tutorial), you will use the *Current Printer Setting*.

2. Make sure the **Orientation** is set to *Portrait*.



3. Skip the **Width** and **Height** fields.

   These fields allow you to specify custom paper sizes.

4. In the **Title Page** field, type *Student Telephone List*.

   This will create a cover page for your report. If you leave this blank, the wizard creates no title page.

5. Select *Horizontal* from the **Field Separators** drop-down list.

   This will place horizontal lines under each row of fields to make it easier to read.

6. Skip the **Group Indent** field.

   This specifies the amount of indentation for each group level of the report. For this report, there are no group breaks.

7. Check the **Page Number** box.

   This adds a page number to the bottom of each page in a Page Footer band.

8. Skip the **Page Margins** section.

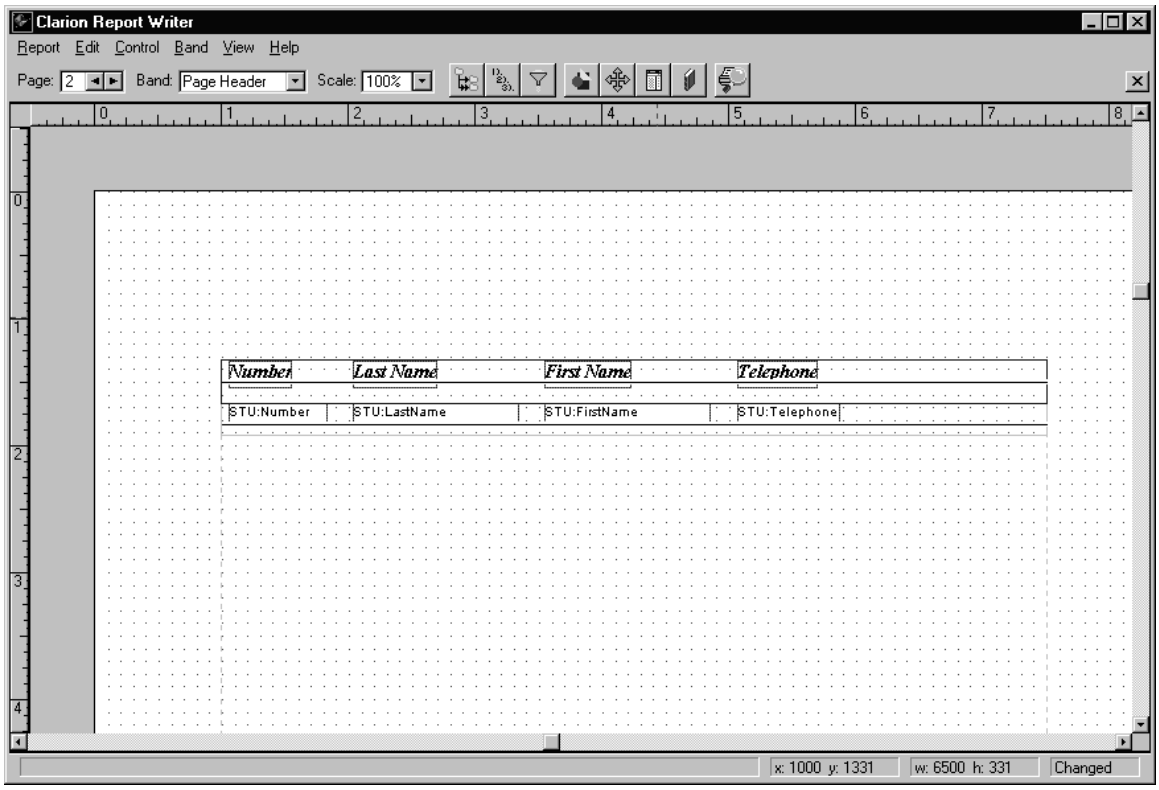
   This sets the margins for report. For this tutorial, the default values are adequate.

9. Press the **Finish** button.

The Report Wizard process the information you supplied, creates the report, and opens the Report Formatter.

The Report Formatter

You design the report's layout in the formatter. This screen features a WYSIWYG (What You See Is What You Get) format. The printed report looks similar to the formatter screen.



You access the Report Formatter's tools through buttons on its toolbar or its menu

Many of the tools have shortcut "hot" keys, displayed to the right of the menu selection. If you begin by using the menus, you can learn the shortcut keys as you go. Remember that if you forget the shortcut key, you can look it up on the menu anytime.

All of the formatter's tools are fully described in Chapter 8—Creating Reports. You will also use most of them later in this tutorial.

Testing the Report

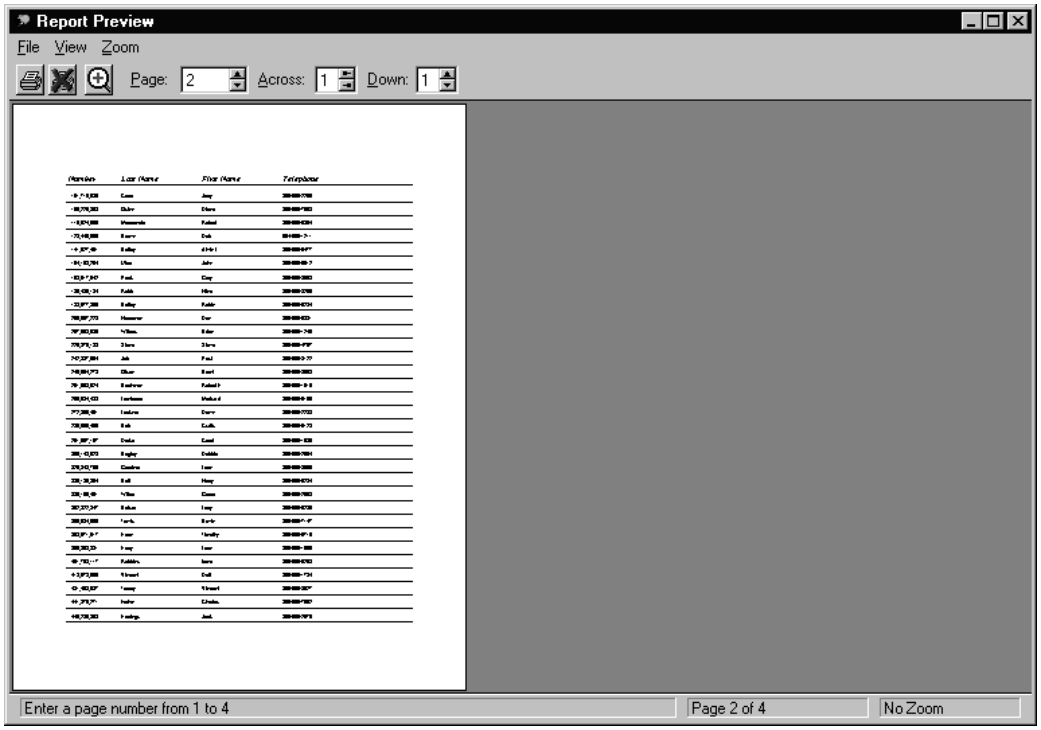
You should do a trial run now to see what you have done so far.

1. Press the  button on the toolbar (or choose **Report** ▸ **Preview**) to run the report in Preview Mode.

   The *Report Preview* appears, displaying the first page of your report.

2. Use the **Page** spin control on the toolbar to turn to page 2.

   The *Report Preview* now displays the second page of your report. This is where the data begins.



The report contains all the information in an adequate format, but we can improve it with a little effort. There are two things we can improve:

- The format of the Student Number. This is actually their Social Security Number, so it should be formatted to print in that format (e.g., 999-99-9999).
- The Last Names are separated from the First Names.

You can easily modify the report to add these cosmetic changes. Exit from the *Report Preview* to edit the report.

3. Press the  button to exit the *Report Preview*. This returns you to the formatter, where you can edit the report.

Editing a Report

Task One: Changing the Format of a Data Field

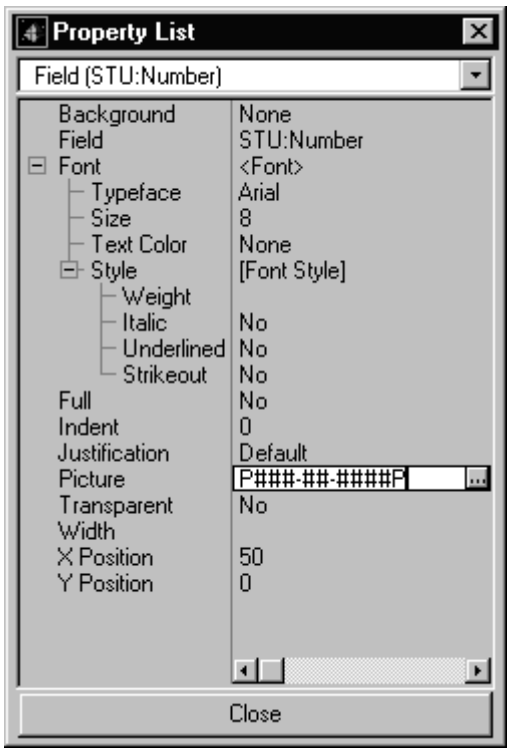
Begin by formatting the Student Number.

In the Report Formatter:

1. RIGHT-CLICK on the *STU:Number* control, then choose **Properties** from the popup menu.

The *Property List* appears.

2. CLICK on the **Picture** entry box (not its ellipsis button), then type *P###-##-####P*, as shown below.



This formats the Student Number as a Social Security number.

3. Press the **Close** Button on the *Property List*.

Task Two: Deleting Fields and Controls from a report

Next, delete the controls for the Student First and Last Name and their associated column headers.

1. CLICK on the *STU:LastName* control, then press DELETE (or RIGHT-CLICK on the control and choose **Delete** from the popup menu).

2. CLICK on the *STU:FirstName* control, then press DELETE (or RIGHT-CLICK on the control and choose **Delete** from the popup menu).

   This removes the two fields that you will replace with the computed field. Next, you will delete one Column heading and modify the other.

3. On the Page Header Band, CLICK on the string control that says "*First Name*," then press DELETE.

4. On the Page Header Band, DOUBLE-CLICK on the string control that says "*Last Name*" control. This enables in-place editing. Edit it to read "*Name*."

5. Press ENTER to accept your changes.

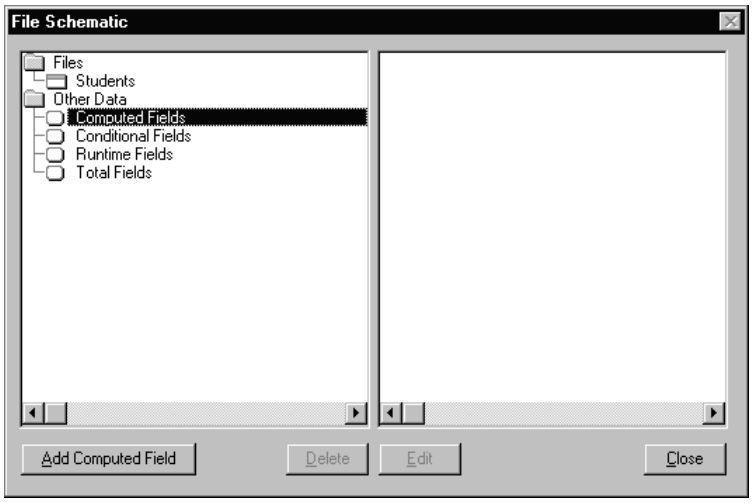
   Since the computed field prints each Student's full name, you only need the single column heading.

Task three: Creating a Computed Field

Next create a computed field that removes the trailing spaces from the Last Name, then add a comma, a space, and the first name.

1. Press the  button on the toolbar (or choose **Report** ▸ **File Schematic**).

   The *File Schematic* window appears. This is the report's file schematic that you specified in the wizard when creating the report.



2. Highlight *Computed Fields*, then press the **Add Computed Field** button. The *Computed Field* window appears.

3. In the **Label** entry box, type *FullName*.

   This provides a unique label that ReportWriter uses to reference the computed field.

4. In the **Description** entry box, type *LastName, FirstName*.

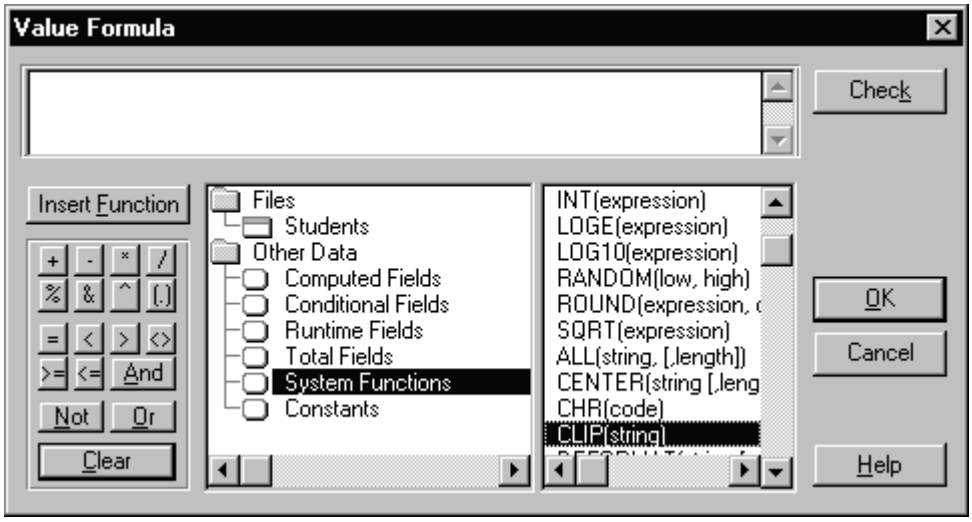
   Although this is not required, a description helps you to identify the computed field. This description appears in the list of computed fields.

5. In the **Picture** entry box, type *@s40* (or press the ellipsis button to use the Picture Editor).

   This formats the computed field to print up to 40 characters.

6. Press the **Formula** button to use the Formula Editor to create the Computed field's formula.

   The Formula Editor appears. The Formula Editor helps you to quickly generate a syntactically correct expression.



7. Highlight *System Functions* in the list box on the left, then highlight *CLIP(string)* in the list box on the right, then press the **Insert Function** button.

   The CLIP function removes trailing spaces from a field.

8. Highlight *Students* in the list box on the left, then highlight *LastName* in the list box on the right, then press the **Insert Field** button.

   These two steps remove the trailing spaces from the LastName field.

9. Press the END key to move the cursor to the end of the expression, then press the **&** button on the Formula Editor keypad.

   This inserts the concatenation operator into the expression to append one string to another.

10. Highlight *Constants* in the list box on the left, then highlight *Text...* in the list box on the right, then press the **Insert Constant** button.

    The *Constant Text* dialog appears. This allows you to enter any constant text.

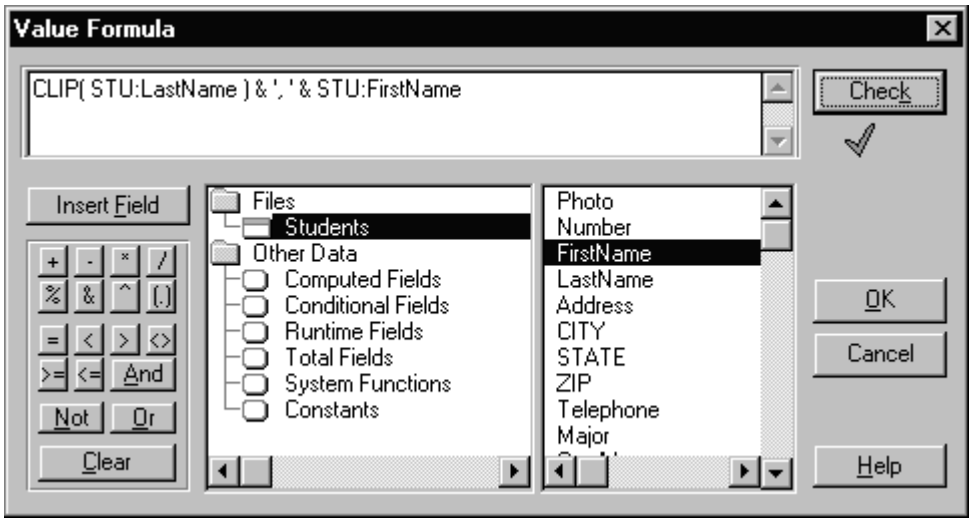
11. Type a comma followed by a space, then press the **OK** button.

    This inserts ', ' into the expression.

12. Press the **&** button on the Formula Editor keypad.

13. Highlight *Students* in the list box on the left, then highlight *FirstName* in the list box on the right, then press the **Insert Field** button.

    This completes the expression. As you become familiar with expression syntax you can simply type in your expression or any portion of it.



14. Press the **Check** button on the Formula Editor to check your expression.

    The Formula Editor displays a green checkmark if the formula syntax is valid, or a red X if the formula syntax is invalid.

15. Press the **OK** buttons on the *Formula Editor* and the *Computed Field* dialogs.

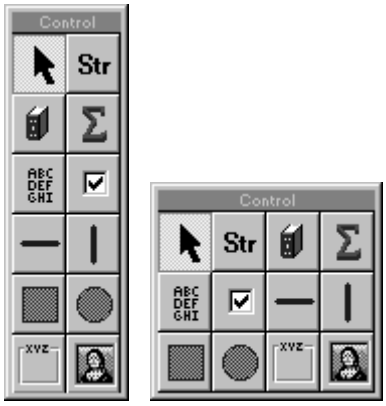
    This returns you to the *File Schematic*. The computed field you just created now appears.

16. Press the **Close** button on the *File Schematic* window.

Task Four: Populating a Computed Field

1. CLICK on the  button on the toolbar to open the **Control Toolbox** (or choose **View** ▸ **Control ToolBox**).

   The Control Toolbox appears. This "floating" toolbox allows you to quickly place controls on your report. ReportWriter's toolboxes are adjustable—you can resize and reposition them by clicking-and-dragging any edge or corner of the toolbox.



2. CLICK on the  button on the toolbox to place a Data Field on the report.

   The *Insert Data Field* window appears. This is the report's file schematic.

3. Highlight *Computed Fields* in the list box on the left, then highlight *FullName* in the list box on the right, then press the **Select** button.

4. CLICK on the location of the report where you want to place the field.

   The point of the Arrow on the mouse cursor positions the upper left corner of the field. This should be in the location formerly occupied by the *STU:LastName* field.

   Your report is complete. You can preview the report to see the result.

Preview the report

1. Press the  button on the toolbar (or choose **Report** ▸ **Preview**) to run the report in Preview Mode.

   The *Report Preview* appears, displaying the first page of your report.

2. Use the **Page** spin control on the toolbar to turn pages.

3. When you are finished, press the  button to exit the *Report Preview*.

   This returns you to the Report Formatter.

4. Choose **Report** ▸ **Close** to exit the Report Formatter and return to the Report Library.

   This is a good time to save your work in progress.

5. Press the  button on the toolbar to save the Report Library.

Copying and Modifying a Report

You have created a report to print a student list in Student Number order. The following steps create a report to print a student list in alphabetical order by Last Name. You could make it "from scratch" using what you have learned, or you can copy the existing report and modify it. Any time two reports are similar, the easiest way to create the new report is to copy the first then modify the copy. This is usually called "working smarter" and ReportWriter is designed to help you work smarter, not harder.

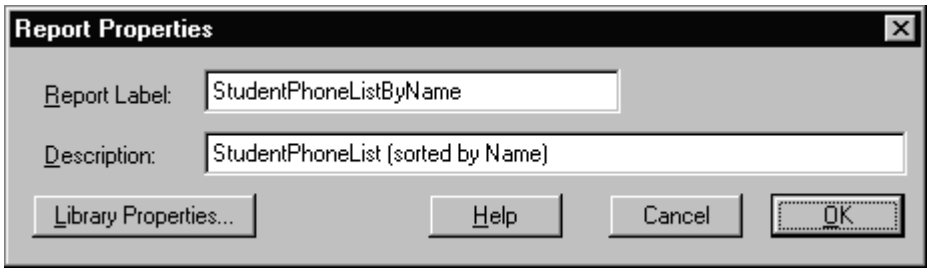
Both reports print all the records from the Student file. The only difference between the two is the order in which the records print.

1. Highlight the report in the **Report Library**, and choose **Report** ▸ **Copy** (or press CTRL+C), then choose **Report** ▸ **Paste** (or press CTRL+V) to paste the copy into the Library.

   ReportWriter copies the report and gives it a unique name. You can modify this name to better describe the report.

2. Press the **Properties** button to rename the new report.

3. Type *StudentPhoneListByName* as the label for the new report.



4. In the **Description** field, type a description for the report.

   Although this is not required, a description helps you identify the report.

5. Press the **OK** button.

   Now you can edit the new report.

6. Highlight the new report and press the **Edit** button.

   The Report Formatter appears.

7. Press the  button on the toolbar or choose **Report** ▸ **Sort Order**.

8. Select *KeyLastName* from the **Order By** drop-down list.

9. Press the **OK** button. This changes the order in which the records are processed. The *KeyLastName* key is defined in last name sequence.

10. Preview the report to see the results. Your report now prints in alphabetical order.

Multi-up Mailing Labels

Follow these steps to design a report to print mailing labels. ReportWriter's Label Wizard simplifies the job and allows you to choose from various types of labels. For this report you will use the same file (STUDENTS.TPS) but you will create a user defined key to sort the records in ZIP code order.

Opening an Existing Report Library

Create this report in the same Report Library as the other reports.

1. If it is not already running, start ReportWriter.

2. If the RWTUTOR Report Library is not open, press the  button on the toolbar (or choose **Report** ▸ **Pick**).

   The Pick List appears displaying recently used Report Libraries.

3. Highlight *RWTUTOR.TXR*, then press the **Select** button. The Report Library opens.

Creating the Multi-up Label Report

Multi-up columns allow two or more records to be printed side-by-side, instead of one below the other. You can use the multi-up column feature to print a report with all or part of the report in multi-column format. By printing records side-by-side, the report uses fewer lines, and can conserve paper and produce a more concise report.

How multi-up columns work

A multi-up column simply has a band defined as a fraction of the paper width. For example, if a band is defined to be half of the paper width, it prints two columns. If it is defined as a third of the paper width, it prints three columns.

In a two column band, records are printed in the following sequence:

1	2
3	4
5	6
7	...

Using the Label Wizard

1. On the *Report Library* window, press the **New** button.

   The *Report Wizard* appears.

2. In the **Report Label** field, provide a unique label for the report.

3. In the **Description** field, type a description for the report.

4. Choose the *Label* style by Selecting the 

button.

5. Specify the Files the report will use (go on to *Specifying the File Schematic*).

Specifying the File Schematic

The File Schematic specifies the file(s) used for the report. This part of the tutorial uses only one file—the STUDENTS file.

1. Press the **Add File** button.

   The Select File window appears. Since the STUDENTS file is the only file in the Report Library it is the only one listed.

2. Highlight *STUDENTS*, then press the **Select** button.

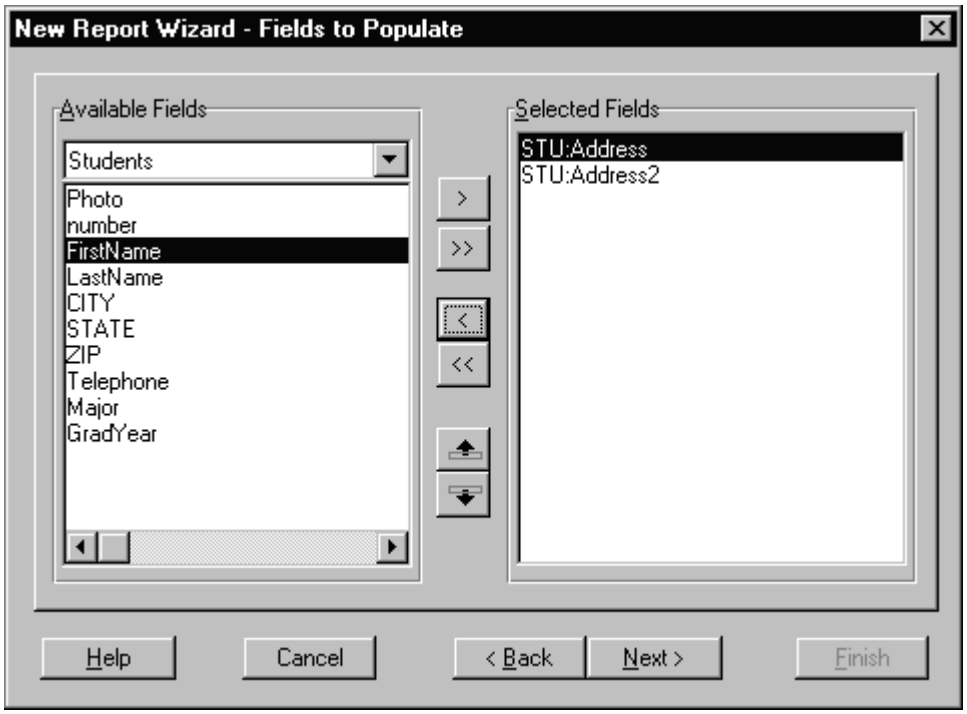
   This places the *Students* file in the file schematic.

3. Press the **Next>** button.

   The *Fields to Populate* window appears.

Populating fields using the Report Wizard

In this dialog, you specify the fields to place on the report. You select fields by adding them to the **Selected Fields** list.



1. Highlight *Address*, then press the > button.

2. Highlight *Address2*, then press the > button.

   The *STU:Address* and *STU:Address2* fields appear in the **Selected Fields** list.

   For now, these are the only fields we want on the report. We will use Computed fields to produce the Student's Name, and the City, State and ZIP Code.

3. Press the **Next>** button.

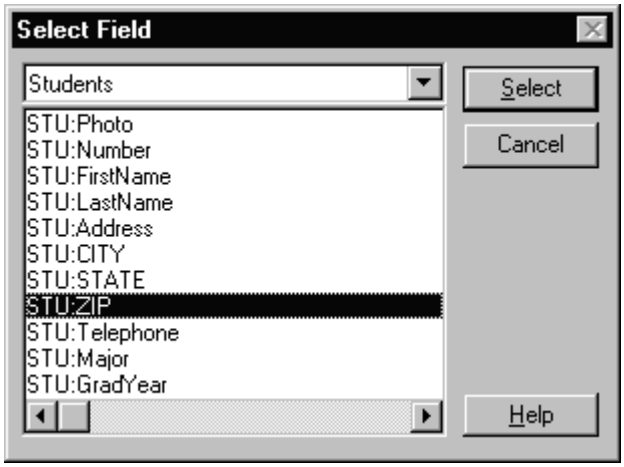
   The Sort Order window appears.

Specifying the Sort Order using the Report Wizard

In this window, you specify the sort order for a report. Notice that the file has no key for ZIP Code. Since you want the labels sorted by ZIP Code, you will define a new user-defined sort order.

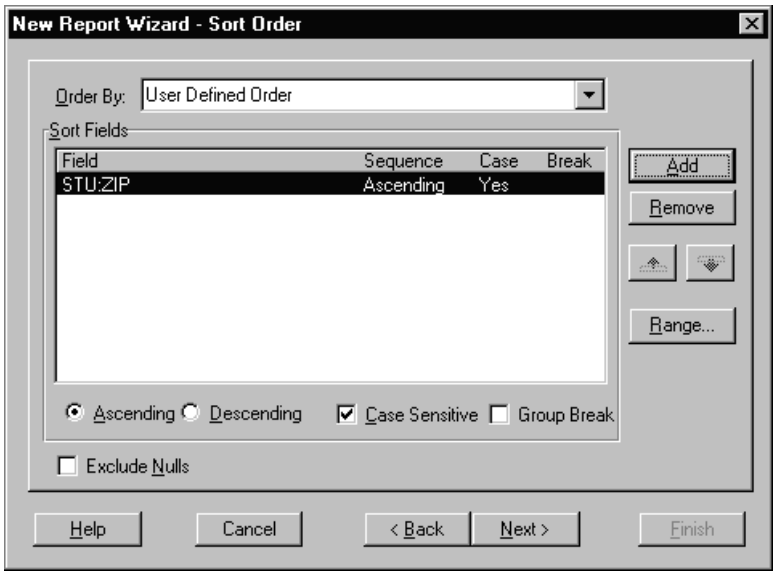
1. Use the **Order By** drop-down list to select *User Defined Order*, then press the **Add** button.

   The *Select Field* window appears.



2. Highlight *STU:ZIP*, then press the **Select** button.

   The ZIP field appears in the **Sort Fields** list.



3. Press the **Next>** button.

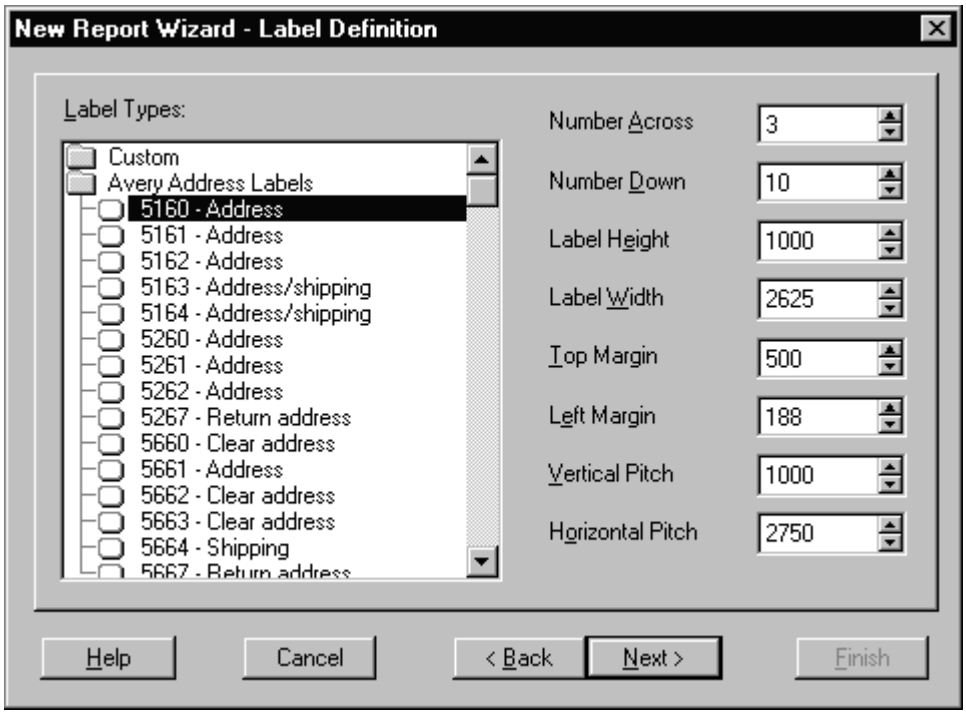
   The *Label Definition* window appears.

Define the Label

The *Label Definition* window allows you to select one of a number of standard labels that are widely available. You can also define your own custom size.

1. Highlight *5160-Address*.

   This specifies Avery 5160 labels (a popular Avery product which has 30 labels per sheet).



2. Press the **Next>** button.

Report Layout

The Report Layout window appears. You'll notice this is different from the Layout window you used to create a Table report.

1. Clear the **Prompts** checkbox.
2. Skip all the other fields (accept the default values).
3. Press the **Finish** button.

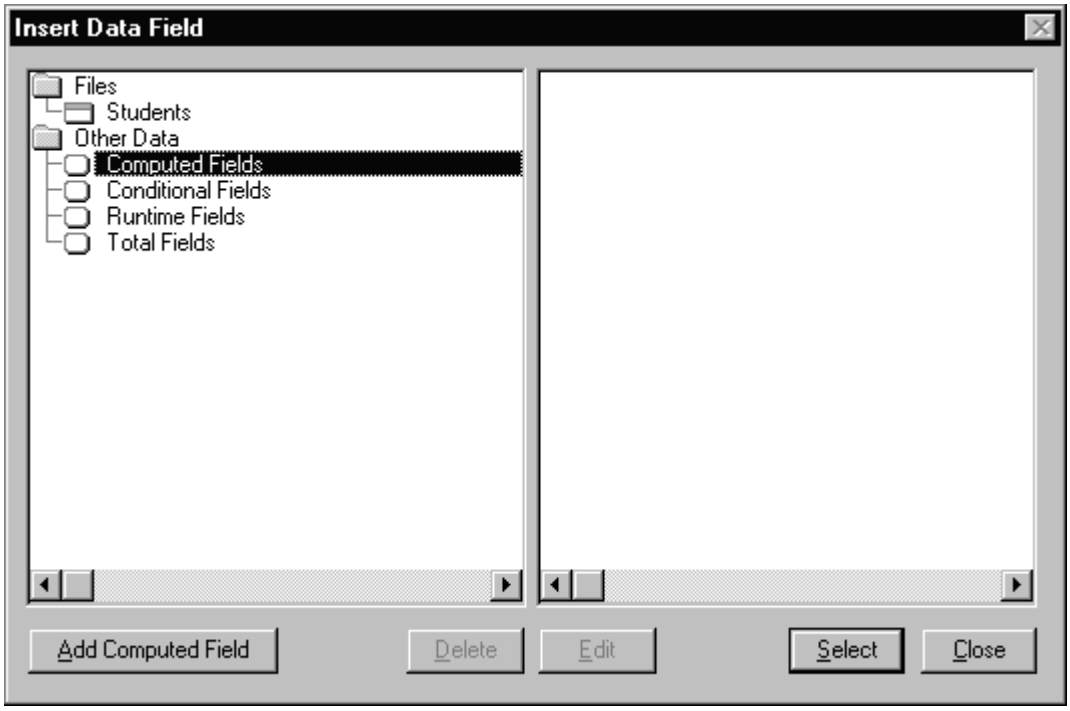
Create the Computed Fields

For the labels, you want two computed fields:

- One to remove trailing spaces from the student's first name, then add a space and the student's last name.
- The other to remove trailing spaces from the student's City, then add a comma, a space, the State, another space, then the ZIPCode.

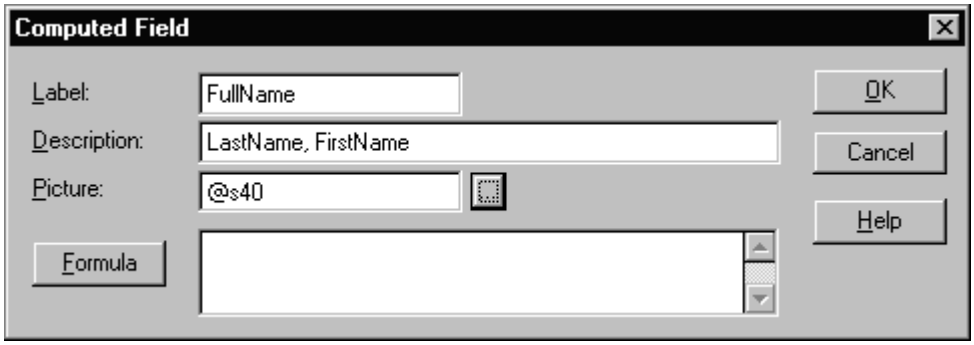
1. Press the  button on the toolbar (or choose **Report** ▸ **File Schematic**).

   The *File Schematic* window appears. This is the report's file schematic that you specified in the wizard when creating the report.



2. Highlight *Computed Fields*, then press the **Add Computed Field** button.

   The *Computed Field* window appears.



3. In the **Label** entry box, type *FullName*.

   This provides a unique label that ReportWriter uses to reference the computed field.

4. In the **Description** entry box, type *FirstName, LastName*.

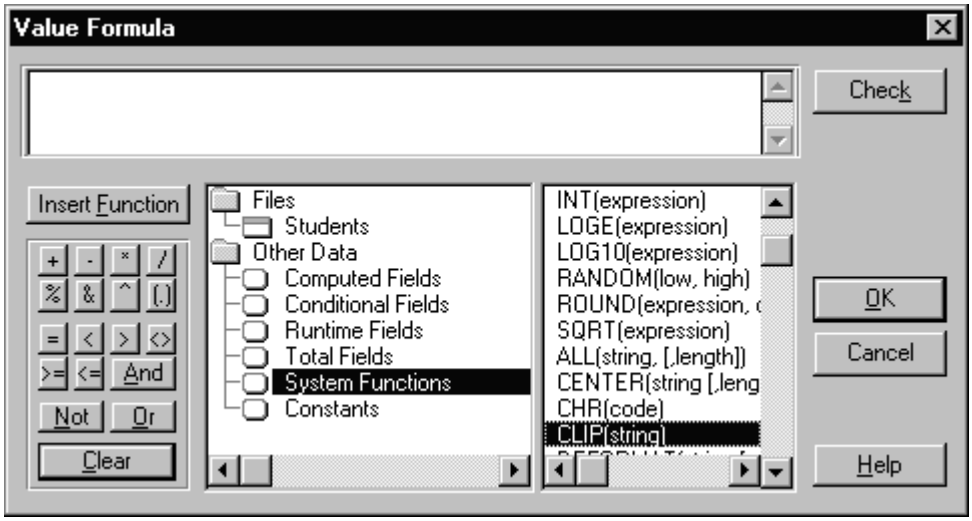
   Although this is not required, a description helps you identify the computed field. This description appears in the list of computed fields.

5. In the **Picture** entry box, type *@s40* (or press the ellipsis button to use the Picture Editor).

   This formats the computed field to print up to 40 characters.

6. Press the **Formula** button to use the Formula Editor to create the Computed field's formula.

   The Formula Editor appears. The Formula Editor helps you quickly generate a syntactically correct expression.



7. Highlight *System Functions* in the list box on the left, then highlight *CLIP(string)* in the list box on the right, then press the **Insert Function** button.

The CLIP function removes trailing spaces.

8. Highlight *Students* in the list box on the left, then highlight *FirstName* in the list box on the right, then press the **Insert Field** button.

   These two steps remove the trailing spaces from the *FirstName* field.

9. Press the END key to move the cursor to the end of the expression, then press the **&** button on the Formula Editor keypad.

   This inserts the concatenation operator into the expression to append one string to another.

10. Highlight *Constants* in the list box on the left, then highlight *Text...* in the list box on the right, then press the **Insert Constant** button.

    The *Constant Text* dialog appears. This allows you to enter any constant text.

11. Type a space, then press the **OK** button.

    This inserts ‘ ‘ into the expression.

12. Press the **&** button on the Formula Editor.

13. Highlight *Students* in the list box on the left, then highlight *LastName* in the list box on the right, then press the **Insert Field** button.

14. Press the **Check** button to verify your expression.

This completes the expression for the name.

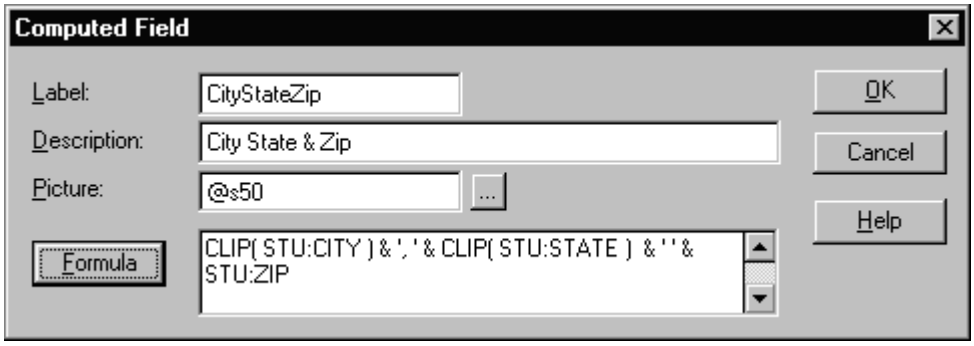
15. Press the **OK** buttons on the Formula Editor and the Computed Field windows.

Create the second Computed Field

This computed field removes trailing spaces from the student's City, then add a space, the State, another space, then the ZIP Code.

1. Create the computed field in the same manner as before.

   Your Computed Field should appear as shown below:



2. After creating the computed fields, press the **Close** button on the *File Schematic* window.

Populate the Computed Fields

Now the computed fields are ready for you to place on the report. For this part of the tutorial, use another method of populating controls.

1. CLICK on the  on the toolbar to open the Field Selection List (or choose **View** ▸ **Field Selection** List). The Field Selection List appears. This "floating" tool allows you to quickly place data fields on your report.

2. Select *Computed Fields* from the drop-down list at the top of the *Field Selection List*.

The Computed Fields you created earlier display on the list.

3. CLICK on *FullName* in the list, then CLICK on the location of the report where you want to place the field.

   You should place the FullName field above the STU:Address field.

4. CLICK on *CityStateZip* in the list, then click on the location of the report where you want to place the field.

   You should place the *CityStateZip* field below the STU:Address field.

   Your report is complete.

5. Preview the report to see the results.

Suppressing the second address line when it is empty

You populated a second address field (STU:Address2). This field will not be used for every address. The letter would still be delivered with a blank line on the label, but it would look better if the line were suppressed. Well, the Label Wizard has already done this for you. You may want to know how it accomplished the line suppression.

1. RIGHT-CLICK on the STU:Address2 field, then choose **Properties**.

   Notice the **Skip if Empty** Property is set to Yes. This specifies that the field will not print if its value is zero or null.

2. CLICK on the **Detail** band, then RIGHT-CLICK on an empty area of it, then choose **Properties**.

Notice the Fixed Height Property is set to Yes. This specifies that the Detail band will always print on a fixed Height. If a Fixed Height were not used, the Detail Band would adjust to the correct height needed depending on the number of controls "skipped."

The combination of these two properties allows you to skip fields and print to a static height area; in other words, it lets you print professional looking labels.

Summary

- A Report Definition File (.TXR) contains definitions for a set of reports.
- A report's layout is defined in bands, which control the printing.
- Fields are deleted by selecting the field and pressing delete.
- A report can be copied in the Report Library by highlighting it and pressing ctrl+c.
- If two reports are similar, it is easier to copy one and modify it than it is to create the new one from scratch.
- Records can be printed side-by-side by using the Label Wizard or by reducing a detail band's width.
- The Picture property in the Property List allows you to change the format of a Data Field.
- The Formula Editor allows you to construct error-free expressions for computed fields.
- Use the Field Selection List or the Data Field tool on the Control Toolbox to populate Data Fields on a report.

5 - TUTORIAL: CREATING COMPLEX REPORTS

Relational Reports

The reports in the first tutorial used a single file. The reports in this tutorial use multiple files, using ReportWriter's Relational capabilities. ReportWriter directly accesses data files in a Relational manner when you specify related files in the File Schematic.

When creating relational reports, a little planning in advance can save a lot of time.

Planning

To start, visualize your report.

- What data will print on the report?
- What files will provide the data?
- Is a user-defined sort order necessary?
- Which records should be grouped together?
- How should it look?

Let's take these questions one-by-one.

What data will print on the report?

In this tutorial, you will create an enrollment report. For this report, you want the Course Description and the Complete Description from the Course file:

```
Course Description
Complete Description
```

Next, the report should show the specific classes for each course. This comes from the Class file:

```
RoomNumber
Scheduled Time
Teacher Name
Class Number
```

Next, the report should contain the information for each student enrolled in the class. This comes from the Enrollment file:

```
Student Number
Student Name
Midterm Grade
FinalExam
TermPaper
```

Finally, the report should also provide:

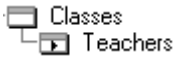
```
A Final Grade for each student
The number of Students in each class
The number of Students in each course
The Average Grade for each Class
```

What files will provide the data?

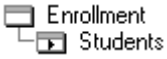
This report uses three related files to define the basic report structure and two lookup files to access more information.

The basic report structure:

In addition, the teacher file allows a lookup through the teacher number link field. This lets you print the teacher's name at the top of each Class.



The Student file allows a lookup through the Student number link field. This lets you print the student's name on each detail (enrollment) band.



Is a user-defined sort order necessary?

The Course file has an established key sorting the records by Course Description. This allows you to create the report of Courses in alphabetical order. The report accesses the class records, sorted by course, allowing the records to be grouped together. The Enrollment file contains a key sorted by Class Number, allowing the records to be grouped together. These keys provide the sort orders necessary for the report.

Which records should be grouped together?

"Group Breaks" place records together in groups based on the value contained in one or more of the record's fields. This gives you a Group Header band and allows you to make a Group Footer band. These bands allow you to print at the beginning or end of each group of records.

In this report, you want all the records for each Course grouped together. The key sorting the Course records by Course Description contains the component we need for the group break (Course Description).

You also want to group Enrollment records for each Class together. In other words, the report accesses the Enrollment records, sorted by Class, breaking on each Class.

A group break occurs each time the Course Description field's value changes. Within that break, another break occurs each time the Class Number changes.

This record grouping goes hand-in-hand with the last question:

How should it look?

You want the report to contain print bands as illustrated here:

```
Header(COU:Description)
   Header(CLA:ClassNumber)
      DETAIL(ENR:StudentNumber)
   Footer(CLA:ClassNumber)
Footer(COU:Description)
```

When you use the Report Wizard and specify these Group Breaks, you'll get a corresponding header for each Group Break. You can also create footers for any of these Group Breaks to produce a report that prints subtotals for each Class and each Course.

In addition, you only want to print classes and courses with students enrolled. To exclude "empty" classes, you will use the Exclude Nulls option.

You have finished the planning stage. You can begin designing the report.

Importing a Data Dictionary

For this tutorial, you will need the entire data dictionary—COLLEGE.TXD. This is a textual representation of a Clarion Data Dictionary.

1. If it's not already running, start ReportWriter.

   If you closed ReportWriter with the first tutorial Report Library open, it appears. If it is already open you can skip the next step.

2. Press the  button, or choose **File** ▸ **Pick**.

3. Highlight *RWTUTOR.TXR*, then press the **Select** button.

   The *Report Library* window appears, listing the reports you have created for this Report Library.

4. Choose **File** ▸ **Import** ▸ **Dictionary**.

   A standard file open dialog appears.

5. Highlight *COLLEGE.TXD*, then press the **OK** button.

   The File definitions and relations are imported. They are now available for reports in this library.

Creating the Relational Report

Now, create the new report.

1. On the *Report Library* window, press the **New** button.

   The Report Wizard appears. This wizard will create the report for you, basing the report layout on what you specify on its screens.

2. In the **Report Label** field, provide a unique label (name) for the report.

   Each report must have a unique label. This identifies the report to the print engine when previewing or printing a report. Labels must begin with a letter and can contain any combination of letters, numbers, the underscore character (_), or the colon character (:). Blank spaces are not allowed.

3. In the **Description** field, type a description for the report.

   Although this is not required, a description helps you to identify the report. This description appears in the Report Library's list of reports.

4. Choose the *Table* style by pressing the 

button.

5. Specify the Files the report uses (go on to Specifying the File Schematic).

Specifying the File Schematic

The files you need to create this report are: *COURSES*, *CLASSES*, and *ENROLLMENT* (you'll add the *TEACHERS* and *STUDENTS* files later). Specify these files in the schematic.

1. Press the **Add File** button.

   The *Select File* window appears. Since you imported the data dictionary, several files now appear.

2. Highlight *COURSES*, then press the **Select** button.

This places the *COURSES* file in the file schematic.

3. With *COURSES* highlighted, press the **Add File** button.

   The *Select File* window appears, this time displaying two sections: Related Files and User Defined Related Files. This allows you to select a related file, or select another for which you can create an ad hoc (user-defined) relation. User Defined Relations are covered in the next chapter.

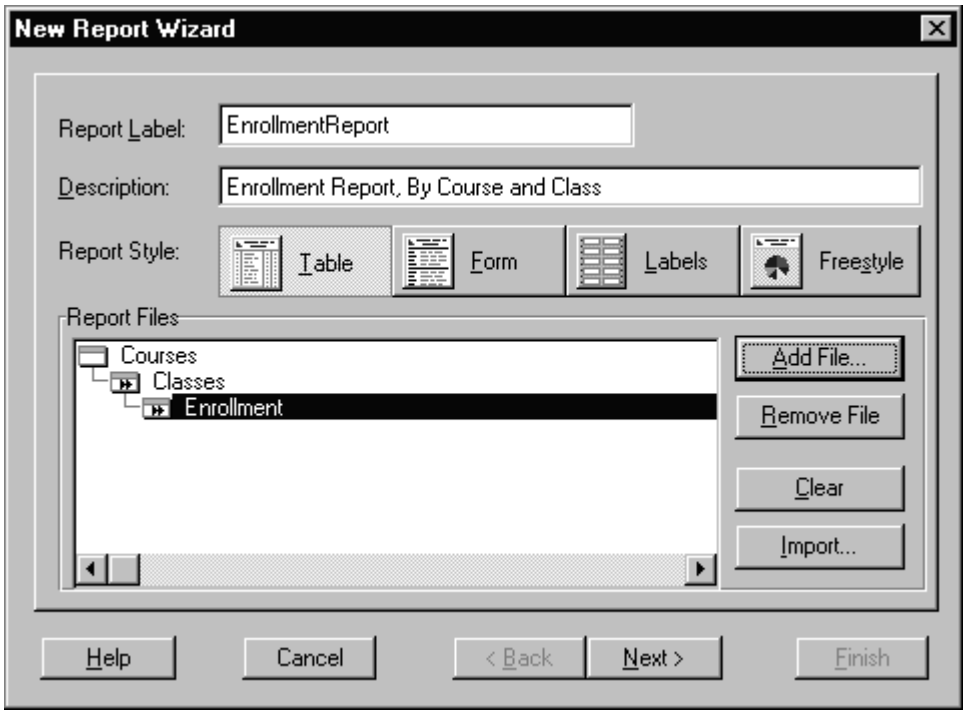
4. Highlight the *CLASSES* file that is attached to the *Related Files* branch, then press the **Select** button.

   This places the *CLASSES* file in the file schematic. Notice it is attached to the *COURSES* file by a double arrow. This indicates that it is a One-to-Many relationship—*CLASSES* is a Child file of the *COURSES* file.

5. With *CLASSES* highlighted, press the **Add File** button.

   The *Select File* window appears, this time displaying the Related Files for the *CLASSES* file.

6. Highlight the *ENROLLMENT* file that is attached to the *Related Files* branch, then press the **Select** button.



   The File Schematic is complete.

7. Press the **Next>** button.

The *Fields to Populate* window appears.

Populating Fields

In this dialog, you specify the fields to place on the report. You select fields by adding them to the **Selected Fields** list. The drop-down list above the **Available Fields** list allows you to select the file in the schematic from which you want to populate fields.

1. Make sure the *Courses* file is selected in the drop-down list.

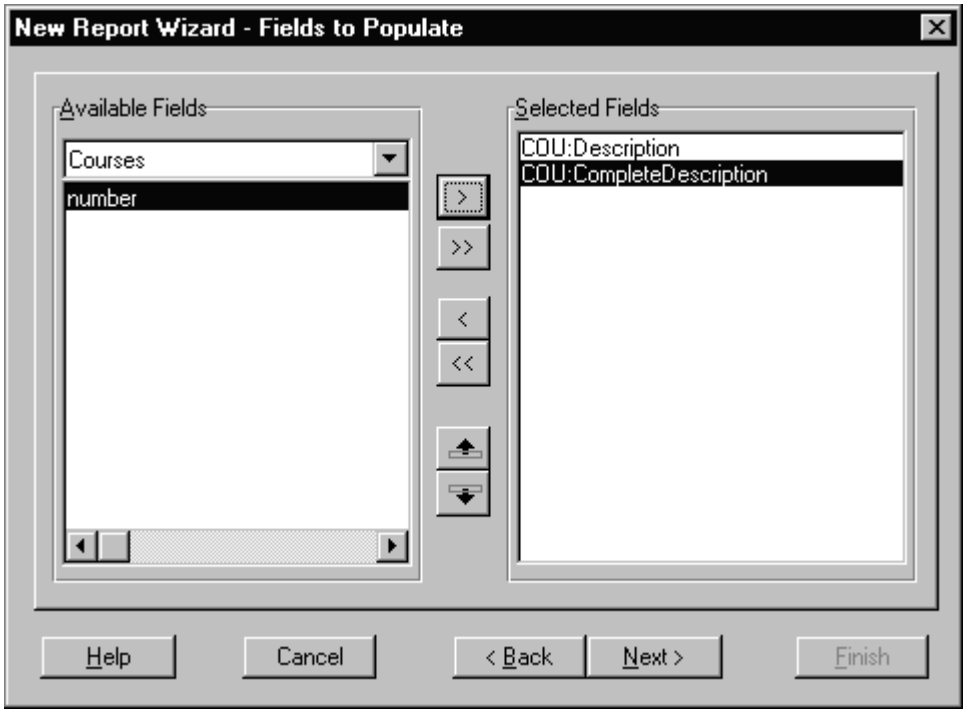
2. In the **Available Fields** list, highlight *Description*, and press the > button.

   The *COU:Description* field appears in the **Selected Fields** list.

3. In the **Available Fields** list, highlight *CompleteDescription*, then press the > button.

   The *COU:CompleteDescription* field also appears in the **Selected Fields** list.

4. Use the drop-down list above the **Available Files** list to select the *Classes* file.



5. Press the >> button to populate all the fields in the *Classes* file.

   All the fields in the *Classes* file now appear in the **Selected Fields** list.

6. In the **Selected Fields** list, highlight *CLA:CourseNumber*, then press the < button.

This removes the *CLA:CourseNumber* from the **Selected Fields** list. Since this is a linking field, you don't need it on the report.

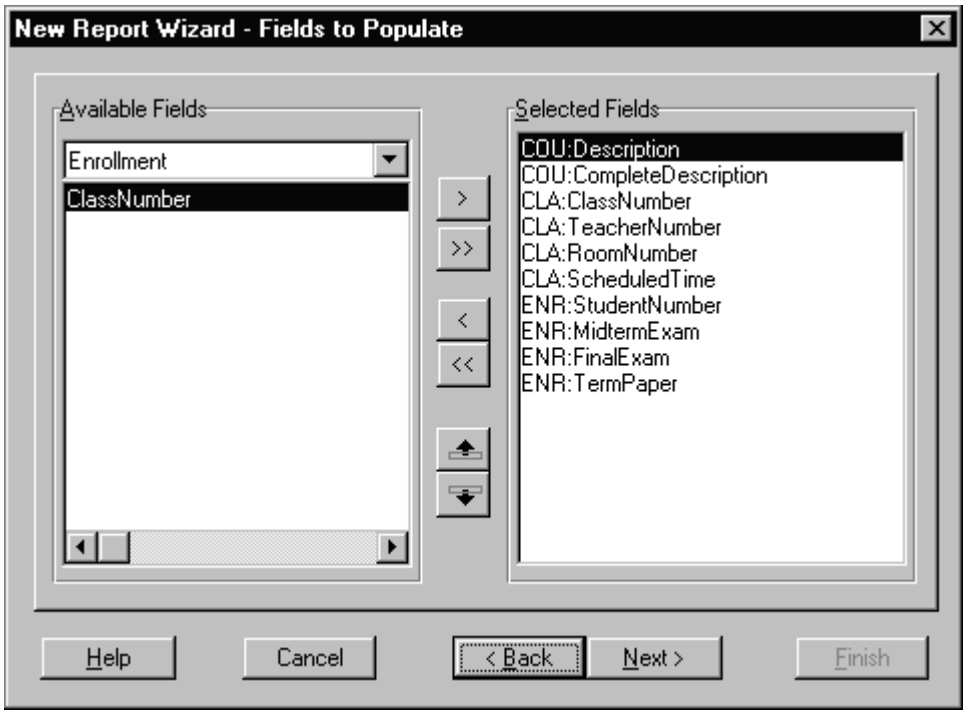
7. Use the drop-down list above the **Available Files** list to select the *Enrollment* file.

8. Press the >> button to populate all the fields in the *Enrollment* file.

   All the fields in the Enrollment file now appear in the Selected Fields list.

9. In the **Selected Fields** list, highlight *ENR:ClassNumber*, then press the < button.

   This removes the *CLA:ClassNumber* from the **Selected Fields** list. Since this is a linking field, you don't need it on the report. The *Fields to Populate* should look like the illustration below:



10. Press the **Next>** button.

    The *Sort Order* window appears.

Specifying the Sort Order and Group Breaks

In this window, you specify the sort order and group breaks for a report. You can sort your report using an existing Key or define a new user-defined sort order. For this report you want to access records in CourseDescription order.

1. Use the **Order By** drop-down list to select *KeyDescription*.

   This key sorts the report by Course's Description.

2. Highlight *COU:Description*, then check the **Group Break** box.

   This will group all related records for a Course together.

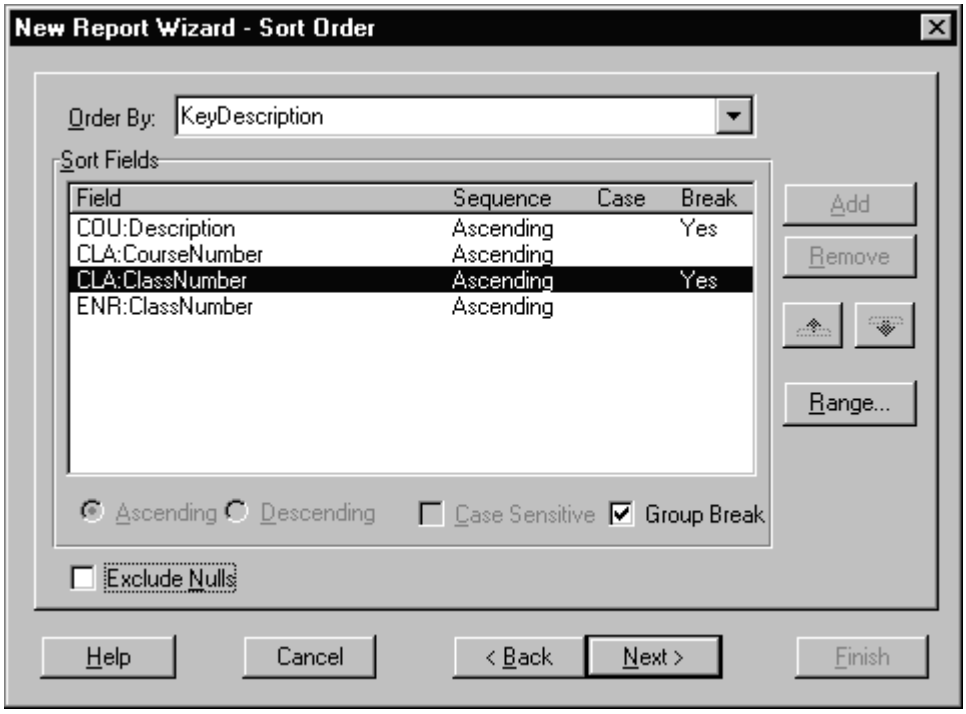
3. Highlight *CLA:ClassNumber*, then check the **Group Break** box.

   This will group all related records for a Class together.

4. Do not check the **Exclude Nulls** box.

   Exclude Nulls specifies that only records which are not zero or blank are considered. Since you want to print Courses and Classes with or without student enrollments, you should include records.

   The *Sort Order* window should appear as pictured below:



5. Press the **Next>** button.

   The *Report Layout* window appears.

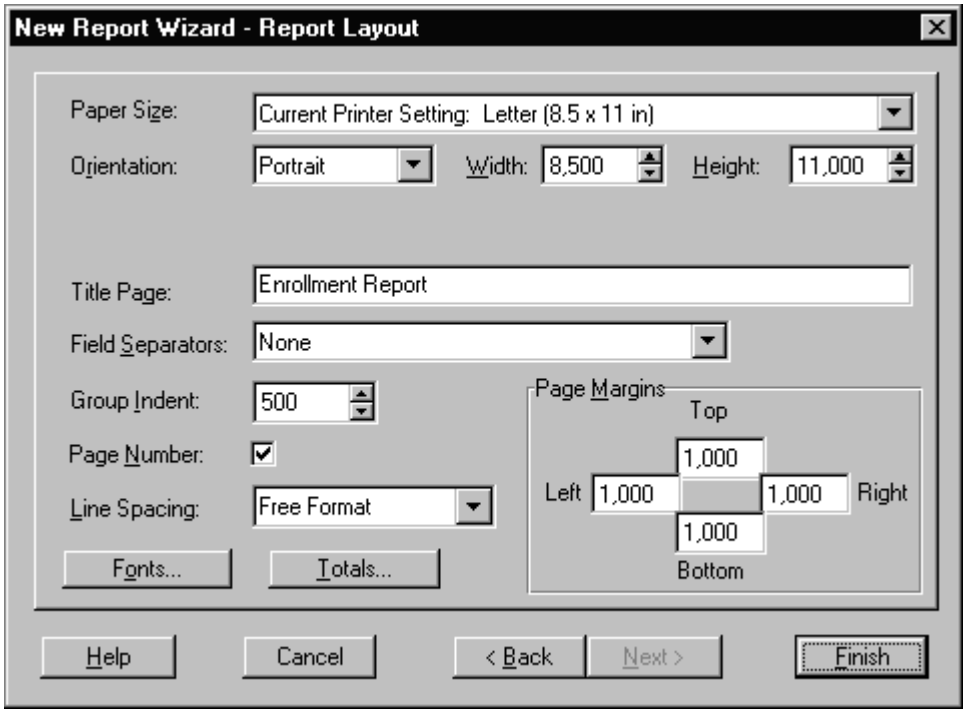
Specify the Report Layout

This window lets you specify a number of options to fine-tune your report's layout. For this tutorial, you will use most of the default settings.

1. Skip the **Paper Size** field.

   This field lets you specify one of a number of common paper sizes. In most cases (and for this tutorial), you will use the *Current Printer Setting*.

2. Make sure the Orientation is set to *Portrait*.



3. Skip the **Width** and **Height** fields.

   These fields allow you to specify custom paper sizes.

4. In the **Title Page** field, type *Enrollment Report*.

   This creates a cover page for your report. If you leave this blank, the wizard creates no title page.

5. Select *Horizontal* from the Field Separators drop-down list.

   This will place horizontal lines under each row of fields to make it easier to read.

6. In the **Group Indent** field, specify *500*.

This specifies the amount of indentation for each group level of the report. For this report, there are two group breaks, so each level is indented 500/1000ths of an inch (i.e., one-half inch).

7. Check the **Page Number** box.

   This adds a page number to the bottom of each page in a Page Footer band.

8. Skip the **Page Margins** section.

   This sets the margins for report. For this tutorial, the default values are adequate.

9. Press the **Finish** button.

   The Report Wizard process the information you supplied, creates the report, and opens the Report Formatter.

   At this point, you can press the  button to preview the report. When you are finished previewing, press the  button to return to the Report Formatter.

Conditional Detail Printing

You can use conditional detail bands to print a band only when a specific condition is true. If you previewed the report, you may have noticed blank records printing in the detail band. Adding a condition to the band can suppress those blank details.

In the Detail Band:

1. RIGHT-CLICK on the Detail band, then select **Properties**.

   Make sure you RIGHT-CLICK on an unused portion of the band.

2. In the Property List, press the ellipsis (...) button next to **Condition**.

   The *Formula Editor* appears.

3. Press the **NOT** button on the Formula Editor keypad.

4. Highlight *System Functions* in the list box on the left, then highlight *EMPTY('filename')* in the list box on the right, then press the **Insert Function** button.

   The *EMPTY* function determines if a record set is empty. For example, when no enrollment records exist for a class. When you select this function, a *Select File* window appears.

5. Highlight *Enrollment* in the list box, then press the **Select** button.

6. Press the **OK** button.

7. Press the **Close** button on the **Property List**.

Variable Length Memos on Reports

The *COU:Description* has a MEMO field populated on it. This is a MEMO field in the data file. Memo fields are generally used to store data of varying lengths. One record may contain several lines of text in its Memo, while another may contain a few words. Typically, you will want your report to adjust to the size of the Memo. If it were a fixed size, you would have blank areas on the report.

The Report Wizard automatically created the Group Header Band to handle the varying length of the Memo Field. When you previewed the report, you probably noticed that some course descriptions were long and others were brief. The report printed with no noticeable "gaps."

This is accomplished with a combination of two property settings. First the TEXT control for the Memo field is set to **Auto Expand**. This specifies that the Text control expands to accommodate all the data in the memo. The second property is on the Group Header Band. It is not set to a **Fixed Height**. This allows the band to expand and contract to accommodate the size of the controls within the band.

To examine the properties used to produce this effect:

1. RIGHT-CLICK on *COU:CompleteDescription*, then select **Properties**.

   Notice **Auto Expand** is set to *Yes*.

2. Press the **Close** button.

3. RIGHT-CLICK on the *COU:Description* Group Header Band, then select **Properties**.

   Notice **Fixed Height** is set to *No*.

4. Press the **Close** button.

   Using these two properties, you can create expanding Memos and Bands in any report.

Creating Group Footers with Total Fields

Before total fields can be placed on a report, they must be defined. Total fields are report-specific. This means they are specific to one report and are calculated at runtime. There are two total fields you want for this report: One to count the number of students in a Course, the other to count the total number of students in a Class. These two total fields are similar. The only difference is the point where each resets to zero. One resets after each Class, the other resets after each Course.

Create the Group Footers

1. Choose **Band** ▸ **Group Footer**, then select *COU:Description*.

   This creates a Group Footer band for *COU:Description* (the group break specified on the Course Description).

2. Choose **Band** ▸ **Group Footer**, then select *CLA:ClassNumber*.

   This creates a Group Footer band for *CLA:ClassNumber* (the group break specified on the Class Number).

Create the Total Fields

Next create total fields to count the number of students in each class and the number of students in each Course.

1. If the Control Toolbox is not active, press the  button on the toolbar.

   This displays the Control Toolbox.

2. Press the  button on the Control Toolbox.

   The *Field Total* window appears. This is where you define the total field.

3. Use the drop-down list at the top of the Field List to select the **Enrollment** file.

4. Highlight *StudentNumber* in the **Fields** list box.

   This specifies that the Student Number from each Enrollment record is used for the calculation.

5. In the **Total Type** group, mark the **Count** radio button.

   This specifies that the total field will count Student Numbers.

6. In the **Picture** entry box, type *@N5* (or press the ellipsis button to use the Picture Editor).

   This specifies the total field will print in a four-digit format, with a comma separating the thousands column (e.g., 9,999).

7. Use the drop-down list next to the **Tally** field to select *ALL*.

   The **Tally** on *All* specifies that every occurrence of *ENR:StudentNumber* is considered.

8. Use the drop-down list next to the **Reset On** field to select *COU:Description*.

   This specifies that the total field resets to zero when the group break for a new Course occurs.

9. Press the **OK** button to return to the formatter.

   Notice the mouse cursor has "turned into" an ABC cursor. That indicates it is "holding" a control to place on the report. When you click on any band of the report, the control is placed at that spot.

10. Inside the *COU:Description* footer band, click on the location where you want to place the total field.

Create another total field to count the students in each class

1. Press the  button on the Control Toolbox.

   The *Field Total* window appears allowing you to define the total field.

2. Use the drop-down list at the top of the Field List to select the *Enrollment* file.

3. Highlight *StudentNumber* in the **Fields** list box.

   This specifies that the Student Number from each Enrollment record is used for the calculation.

4. In the **Total Type** group, mark the **Count** radio button.

   This specifies the total field will count Student Numbers.

5. In the **Picture** entry box, type *@N5* (or press the ellipsis button to use the Picture Editor).

   This specifies the total field will print in a four-digit format, with a comma separating the thousands column (e.g., 9,999).

6. Use the drop-down list next to the **Tally** field to select *ALL*.

   The **Tally** on *All* specifies that every occurrence of *ENR:StudentNumber* is considered.

7. Use the drop-down list next to the **Reset On** field to select *CLA:ClassNumber* band.

   This specifies that the total field resets to zero when the group break for a new Course occurs.

8. Press the **OK** button to return to the formatter.

   Again the mouse cursor has "turned into" an ABC cursor, indicating it is "holding" a control to place on the report. When you click on any band of the report, the control is placed at that spot.

9. Inside the *CLA:ClassNumber* footer band, click on the location where you want to place the total field.

Populating String Controls to Identify the Totals

You'll want to identify these total fields by printing some text. A simple String control with some text will suffice.

1. Press the Str button on the Control Toolbox.

   Again the mouse cursor has "turned into" an ABC cursor, indicating it is "holding" a control to place on the report. When you click on any band of the report, the control is placed at that spot.

2. Inside the *CLA:ClassNumber* footer band, CLICK on the location (to the right of the total field) where you want to place the total field.

   When you place the control, it is in Edit Mode. In Edit Mode you can edit the text directly. If you later want to edit the text, you can DOUBLE-CLICK on the control.

3. Inside the String Control, type *Students Enrolled*, then press ENTER.

   The String control you want for the other total field is exactly the same, so you can use ReportWriter's copy and paste feature.

4. CLICK on the *Students Enrolled* String Control, then choose **Edit** ▸ **Copy** (or press CTRL+C).

5. CLICK on the *COU:Description* footer band, then choose **Edit** ▸ **Paste** (or press CTRL+V).

6. Position the control (to the right of the total field) by dragging it with the mouse or pressing the cursor arrow keys.

Adding Lookup Fields

Next you will add some fields from "lookup" files to provide more understandable data. For example, for each Class you are printing a Teacher Number that uniquely identifies a teacher. But it will be more understandable to print the teachers' names on the report.

Adding the Lookup files to the File Schematic

1. Press the  button on the toolbar.

   This displays the File Schematic.

2. With *Classes* highlighted, press the **Add File** button.

   The *Select File* window appears, displaying two sections: Related Files and User Defined Related Files.

3. Highlight the *Teachers* file that is attached to the *Related Files* branch, then press the Select button.

   This places the *Teachers* file in the file schematic. Notice it is attached to the *Classes* file by a single arrow. This indicates that it is a Many-to-One relationship—*Teachers* is a Parent file of the *Classes* file.

Next add the *Students* file.

4. With *Enrollment* highlighted, press the **Add File** button.

   The *Select File* window appears, displaying two sections: Related Files and User Defined Related Files.

5. Highlight the *Students* file that is attached to the *Related Files* branch, then press the **Select** button.

   This places the *Students* file in the file schematic. Notice it is attached to the *Enrollment* file by a single arrow, indicating it is a Many-to-One relationship—*Students* is a Parent file of the *Enrollment* file.

6. Press the **Close** button to close the File Schematic.

Populating fields from the Lookup files

These data fields are populated in the same manner as any other field.

1. CLICK on the  on the toolbar to open the Field Selection List (or choose **View ▸ Field Selection List**).

   The *Field Selection List* appears. This "floating" tool allows you to quickly place data fields on your report.

2. Select *Teachers* from the drop-down list on the *Field Selection List*.

   The Fields in the *Teacher* file appear on the list.

3. CLICK on *LastName* in the list, then click on the location of the report where you want to place the field.

   You should place the *LastName* field in the *CLA:ClassNumber* Group Header band.

4. CLICK on *FirstName* in the list, then click on the location of the report where you want to place the field.

   You should also place the *FirstName* field in the *CLA:ClassNumber* Group Header band beneath the *LastName* field.

5. Select *Students* from the drop-down list on the *Field Selection List*.

   The Fields in the *Students* file appear on the list.

6. CLICK on *LastName* in the list, then CLICK on the location of the report where you want to place the field.

   You should place the *LastName* field in the detail band to the left of the *ENR:StudentNumber* field.

   Your report now prints the desired information. If you want to, preview the report to see the results.

Adding a Computed field

There is only one item left from the original plan for the report—printing each student's final grade. The Final Grade is a weighted calculation, based on each individual grade (Midterm Exam, Final Exam, and Term Paper). The developer who designed this database did not store the grade in the student file because it is easily calculated using the data already stored in the file. To store the final grade in the student file would waste space (data redundancy).

This college has a standard formula for computing a student's final grade—the Mid-Term Exam is worth 25%, the Final Exam is worth 25%, and the Term Paper is worth 50%.

To print the final grade, create a Computed field using this formula:

```
(ENR:MidTermExam * 0.25) + (ENR:FinalExam * 0.25) + (ENR:TermPaper * 0.5)
```

Create the Computed Field

Next create the Computed field to calculate the final grade for each student.

1. If the Control Toolbox is not active, press the  button on the toolbar.

   This displays the Control Toolbox.

2. Press the  button on the Control Toolbox.

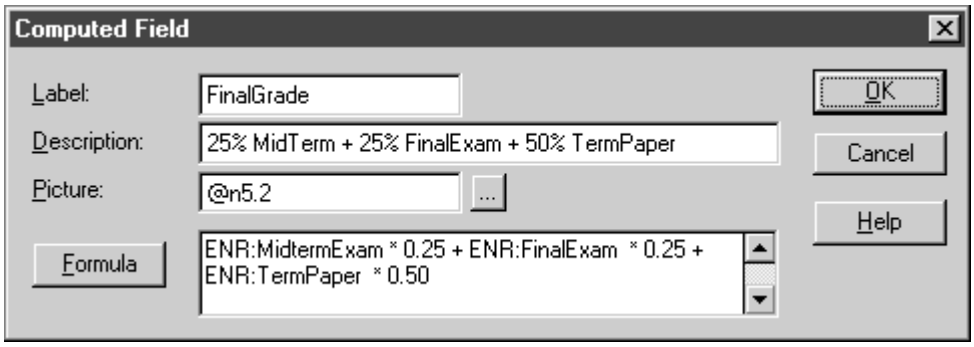
   The *Insert Data Field* window appears. This is the same as the report's File Schematic.

3. Highlight Computed Fields in the list box on the left, then press the **Add Computed Field** button.

   The *Computed Field* window appears.

4. Create the Computed Field in the same manner as before, then press the **OK** button.

   The completed *Computed Field* window should look like the illustration below:



5. In the *Insert Data Field* window, highlight *Computed Fields* in the list box on the left, then highlight *FinalGrade* in the list box on the right, then press the **Select** button.

6. CLICK on the Detail band at the location of the report where you want to place the field (to the right of the *ENR:TermPaper* field).

7. Place a String control in the CLA:ClassNumber Group Header Band to serve as a column header for the Final Grade.

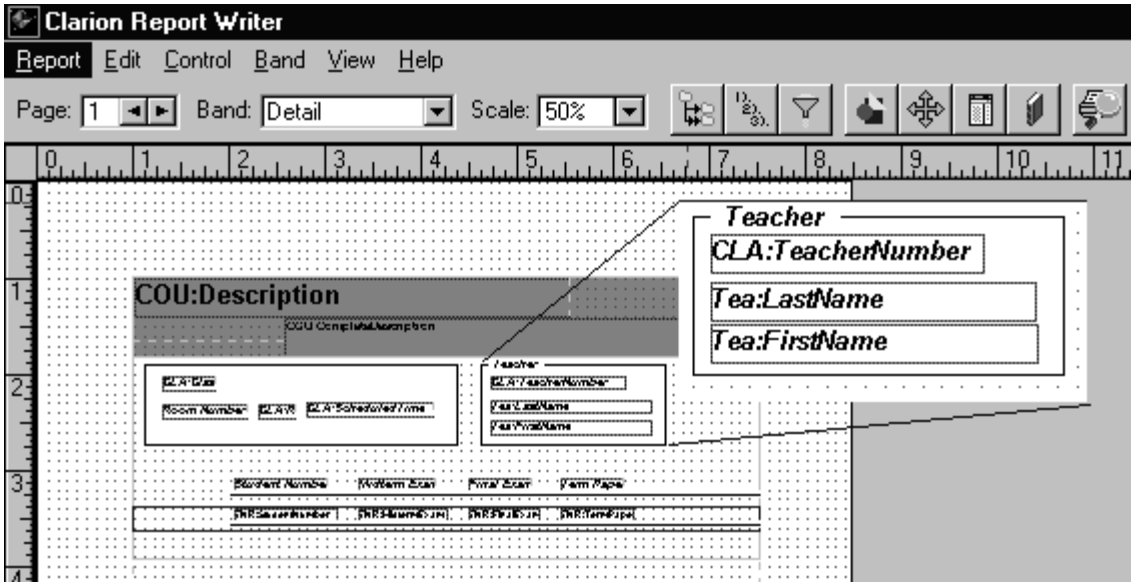
Cosmetic Enhancements for the Report

The report is fully functional now, but you can improve it with little effort using some of ReportWriter's cosmetic features.

Grouping Controls using a Group Box

The GROUP control allows you to reference a group of controls as one entity. When you set properties for the GROUP, all controls within that group inherit the setting unless you explicitly set it for the control. This allows you to modify properties for several controls at once.

Another use for a Group box is to create a box around a group of controls and optionally provide text to identify the group. An example is illustrated below:



For this report, putting fields within a group box can also eliminate the need for prompts next to the Teacher fields. The Group Box text identifies the data.

1. If the Control Toolbox is not active, press the  button on the toolbar.

   This displays the Control Toolbox.

2. Press the  button on the Control Toolbox.

3. CLICK on the *CLA:ClassNumber* Group Header band at the location where you want to place the Group Box (somewhere on the right side).

4. CLICK and DRAG the *Tea:LastName*, *Tea:FirstName*, and *CLA:TeacherNumber* fields into the group box.
5. Resize the group box as needed by CLICKING and DRAGGING its handles.
6. Delete the *Teacher Number:* prompt (String control).
7. RIGHT-CLICK on the group box, and select **Properties**.
8. In the Text property, type *Teacher*, then close the *Property List*.

   The text now appears at the top of the Group Box. Next, create a Group box for the Class information.

Create a Group Box for the Class information

1. Press the  button on the Control Toolbox.
2. CLICK on the *CLA:ClassNumber* Group Header band at the location where you want to place the Group Box (somewhere on the left side).
3. CLICK and DRAG the *CLA:ClassNumber*, *CLA:RoomNumber*, and *CLA:ScheduledTime* fields into the group box.
4. Resize the group box as needed by clicking and dragging its handles.
5. Delete the associated prompts (String controls).
6. RIGHT-CLICK on the group box, and select **Properties**.
7. In the Text property, type *Class*, then close the *Property List*.

The text now appears at the top of the Group Box.

Using Different Fonts

Different font properties can make elements of your report stand out. ReportWriter allows you to specify the font's Typeface, Size, Color, and Style. Use any combination you desire to create the report you want.

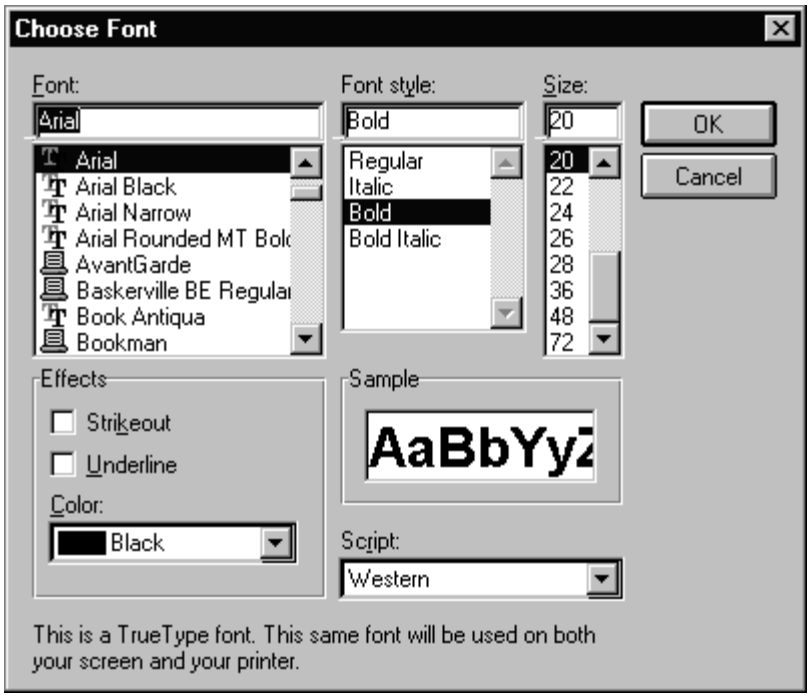
For this report, you want to make the font on a Course Description larger and Bold. That will make a better Group Header and distinguish it from the other elements of the report.

In the COU:Description Header Band:

1. Delete the String Control that says *Description:* and the String Control that says *Complete Description:*

   You won't need the prompts to identify the fields.
2. Move the *COU:Description* field all the way to the left by CLICKING and DRAGGING it with the mouse.
3. RIGHT-CLICK on the *COU:Description* field, then choose **Font**. The *Choose Font* dialog appears. This dialog allows you to modify all of the Font Properties at the same time.

4. Select *Arial* as the Font, *Bold* as the Font Style, and *20* as the Size, as pictured below, then press the **OK** button.



5. Move the *COU:CompleteDescription* field down (so it is not obscured) by CLICKING and DRAGGING it with the mouse.

Using Background colors

You can also use background colors to enhance elements on your report. If you have a color printer, you can print the background colors. On a non-color printer, the colors are converted to greyscale by the printer's driver. For this report, a grey background would be nice on the *COU:Description* Header Band.

In the COU:Description Header Band:

1. RIGHT-CLICK on the *COU:Description* Header band, then select **Properties**.

   Make sure you RIGHT-CLICK on an unused portion of the band.

2. In the Property List, press the ellipsis (...) button next to **Background**.

   A standard windows color dialog appears.

3. CLICK on the grey-colored box below **Basic Colors**, then press the **OK** button.

4. Press the **Close** button on the Property List.

The band now has a grey background color. Controls on a band take on the band's properties unless a property is explicitly set for a control. Therefore, the controls on this band take on the background color of the band.

Controlling Pagination

There are four properties which control the manner in which report elements span pages (pagination). Using these pagination options, you can create a report that prints in an organized manner and ensures that every page is understandable.

Page Before Specifies to print the structure on a new page. To reset the page number to a value you specify, type it in the **Page Before** field. To force a Page Before without resetting the page number, specify -1.

Page After Specifies to print the structure, then force a new page. To reset the page number to a value you specify, type it in the **Page After** field. To force a Page After without resetting the page number, specify -1.

With Prior Prevents 'orphan' elements in a printout. An 'orphaned' print element is one which prints on a following page, separated from its related items. The value specifies the number of preceding elements to print—a value of "1," for example, specifies that the previous element must print on the same page. When placing subtotals or totals in a band, use the WITHPRIOR attribute to insure they print with at least one member of the column above it when a page break occurs.

With Next Prevent 'widow' elements in a printout. A 'widowed' print element is one which prints, but then is separated from the succeeding elements by a page break. The value specifies the number of succeeding elements to print—a value of '1,' for examples, specifies that the next element must print on the same page, else page overflow puts them both on the next.

For this report, use a combination of these properties to control the report's pagination. This time use the floating Property list to set the properties.

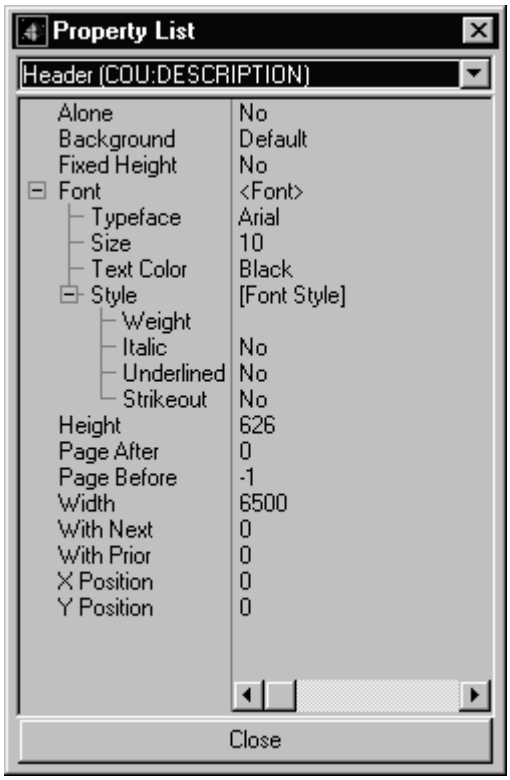
1. CLICK on the  button.

   The floating *Property List* appears. This toolbox allows you to set properties of any currently selected element of your report.

2. CLICK on the COU:Description Header band.

3. In the **Page Before** entry box in the Property List, type *-1*.

   This forces a page break before each group header, ensuring that each new course starts at the top of a page. By specifying -1, page numbers are not reset and continue to increment throughout the report.



4. CLICK on the *COU:Description* Footer band.

5. In the With Prior entry box in the Property List, type *1*.

   This ensures that each course footer prints with at least one prior element of the report (the CLA:ClassNumber footer).

6. CLICK on the CLA:ClassNumber Footer band.

7. In the With Prior entry box in the Property List, type *1*.

   This ensures that each Class footer prints with at least one prior element of the report (the detail band).

The report is complete. Preview the report, then close it.

Congratulations! You have completed the second tutorial.

Summary

- When creating relational reports, a little planning in advance can save a lot of time.
- When you import a Clarion Data Dictionary, ReportWriter imports file relationships defined in the dictionary.
- When defining the file schematic for your report, you add related files to the currently highlighted file.
- The Sort Order dialog allows you to specify the order in which records print and set group breaks to group records together.
- To exclude "empty" records, use the **Exclude Nulls** option.
- The Report Wizard automatically handles the varying length of Memo fields by setting the **Auto Expand** property on the Text control and not setting the **Fixed** property on the band.
- Group Footers print after all Detail bands for a group print.
- Relations allow you to lookup values from a parent file to print on a report.
- Group Boxes allow you to group controls together.
- Background colors and Fonts allow you to highlight elements of your report.
- Control pagination using the **Page Before**, **Page After**, **With Next**, and **With Prior** properties.

6 - TUTORIAL: CREATING USER-DEFINED RELATIONAL REPORTS

User-defined Relational Reports

The reports in the last tutorial used a Clarion Data Dictionary. When you imported the dictionary, ReportWriter imported the relations between files. There may be times when you have a set of related files, but do not have a dictionary defining them. You can create the same reports by joining the related files in user-defined relationships.

In this tutorial, you will use the same files—this time importing them one-by-one and defining the relations in ReportWriter. This will teach you to use related files when you do not have a dictionary available.

Begin by examining the files to use and the fields used to link them in a relationship. When creating a user-defined relationship, the file must have fields which contain identical data to use as a link between them. For example, Enrollment Records have ENR:ClassNumber fields which link to a record in the Class file through the CLA:Number field.

As in the last tutorial, you will create an enrollment report. You want the Course Description and the Complete Description from the Course file:

```
Course Description
Complete Description
```

Next, the report should contain the information for each class of that course from the Class file:

```
RoomNumber
Scheduled Time
Teacher Name
Class Number
```

Next, the report should contain the information for each student enrolled in the class (from the Enrollment file):

```
Student Number
Student Name
Midterm Grade
FinalExam
TermPaper
```

Finally, the report should also provide:

```
A Final Grade for each student
The number of Students in each class
The number of Students in each course
```

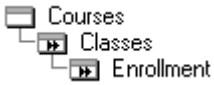
Related Data Files

There are five data files in the tutorial data set which you will use:

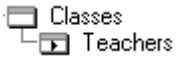
STUDENTS.TPS
TEACHERS.TPS
COURSES.TPS
CLASSES.TPS
ENROLLME.TPS

Three related files to define the basic structure and two lookup files access more information.

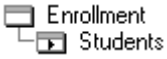
The basic report structure:



In addition, the teacher file allows a lookup through the teacher number link field. This lets you print the teacher's name at the top of each Class.



The Student file allows a lookup through the Student number link field. This lets you print the student's name on each detail (enrollment) band.



The last tutorial used relationships defined in the dictionary. In this report, you will import the individual files and define these relationships yourself.

Creating a Report with Imported Data Files

For this tutorial, create a new Report Library.

1. If it's not already running, start ReportWriter.

   If you closed ReportWriter with a Report Library open, it appears. If it is open, press the **Close** button to close it.

2. Press the  button on the toolbar (or choose **File** ▸ **New**).

   The *New Report Library* window appears. This is where you specify the Report Library's properties.

3. In the **Filename** field, type *RWTUTOR2*.

   ReportWriter automatically adds the .TXR extension for the file.

4. In the **Working Directory** field, type C:\CLARION6\EXAMPLES\RWTUTOR (the path of the working directory) or press the ellipsis (...) button to select it from a standard file dialog.

If you specified a different directory during installation, modify the entry as needed. The working directory is the first directory in which ReportWriter looks for your data files.

5. In the **Description** entry box, type a description for this Report Library.

   Although optional, the description can help you identify the Report Library.

6. Skip the **Password** field.

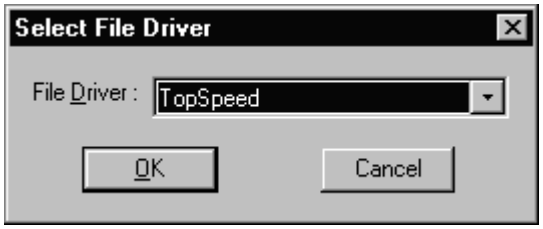
   You can use password protection for a Report Library by specifying a password here. For this tutorial, do not use password protection.

7. In the **Create Using** section, mark the **Database** radio button.

   This specifies that the Report Library is based on a data file instead of a Data Dictionary or Report Library.

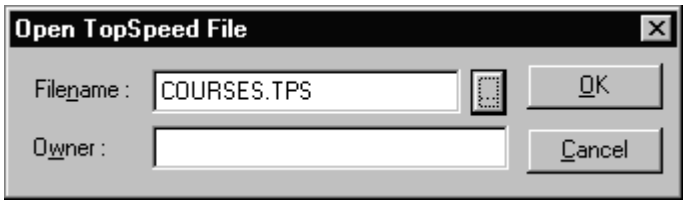
8. Press the ellipsis (…) button next to the **Filename** entry box.

   The *Select File Driver* dialog appears.



9. Select *TOPSPEED* from the File Driver drop-down list, then press the **OK** button.

   The *Open TopSpeed File* window appears.



10. Press the ellipsis (…) button next to the **Filename** entry box.

    A standard file open dialog appears.

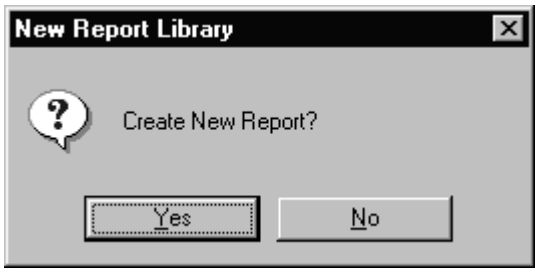
11. Select *COURSES.TPS*, then press the **OK** button (if necessary, “walk” the directory tree to the \CLARION6\EXAMPLES\RWTUTOR subdirectory).

12. Press the **OK** button on the *Open TopSpeed File* window.

    This imports the *COURSES* file. You will import the other files while creating the report.

13. Press the **Create** button to make the Report Library.

Since this Report Library does not yet contain a report, ReportWriter asks if you want to create one now.



14. Press the **Yes** button to create a report.

Creating the Relational Report

The *Report Wizard* appears.

1. In the **Report Label** field, provide a unique label (name) for the report.

   Each report must have a unique label to identify the report to the print engine when previewing or printing a report. Labels must begin with a letter and can contain any combination of letters, numbers, the underscore character (_), or the colon character (:). Spaces are not allowed.

2. In the **Description** field, type a description for the report.

   Although this is not required, a description helps you to identify the report. This description appears in the Report Library's list of reports.

3. Choose the *Table* style by selecting the 

button.

4. Specify the Files the report uses (go on to *Specifying the File Schematic*).

Specifying the File Schematic

The files you need to create this report are: *COURSES*, *CLASSES*, *ENROLLMENT*, *TEACHERS*, and *STUDENTS*. Specify these files in the schematic.

1. Press the **Add File** button.

   The *Select File* window appears. Since you only imported the Courses file, it is the only one to appear.

2. Highlight *COURSES*, then press the **Select** button.

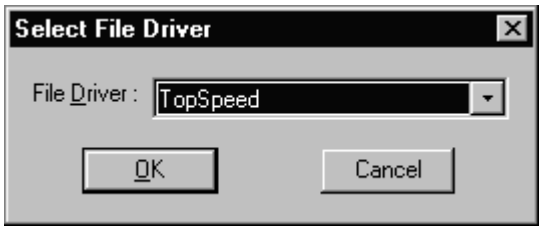
   This places the *COURSES* file in the file schematic.

3. With *COURSES* highlighted, press the **Add File** button.

The *Select File* window appears, this time displaying only one section—User Defined Related Files—because no relations exist. This lets you create an ad hoc (user-defined) relation.

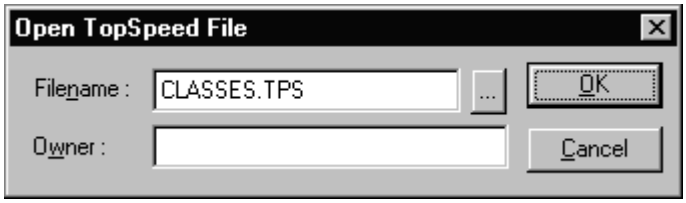
4. Press the **Other File...** button.

   The *Select File Driver* dialog appears.



5. Select *TOPSPEED* from the File Driver drop-down list, then press the **OK** button.

   The *Open TopSpeed File* window appears.



6. Press the ellipsis (…) button next to the **Filename** entry box.

   A standard file open dialog appears.

7. Select *CLASSES.TPS*, then press the **OK** button.

8. Press the **OK** button on the **Open TopSpeed File** window.

   The *User Defined Relationship* window appears. This is where you define the relationship.

9. Mark the **1 to Many** radio button at the bottom of the window.

   This specifies that the files are joined in a One to Many relationship. There are many classes for each course.

10. Press the **Add** button.

    The Link Fields window appears. This is where you specify the fields which link the files together.

11. Select *COU:NUMBER* from the **Link From** drop-down list.

12. Select *CLA:CourseNumber* from the **Link To** drop-down list.

    This specifies that these fields join the files together.

13. Press the **OK** button on the *Link Fields* window, then press the **OK** button on the *User Defined Relationship* window.

    This places the *CLASSES* file in the file schematic. Notice it is attached to the *COURSES* file by a double arrow. This indicates that it is a One-to-Many relationship—*CLASSES* is a Child file of the *COURSES* file.

Adding files to the schematic

1. With *CLASSES* highlighted, press the **Add File** button.

   The *Select File* window appears.

2. Press the **Other File...** button.

   The *Select File Driver* dialog appears.

3. Select *TOPSPEED* from the **File Driver** drop-down list, then press the **OK** button.

   The *Open TopSpeed File* window appears.

4. Press the ellipsis (…) button next to the **Filename** entry box.

   A standard file open dialog appears.

5. Select *ENROLLME.TPS*, then press the **OK** button.

6. Press the **OK** button on the *Open TopSpeed File* window.

   The *User Defined Relationship* window appears. This is where you define the relationship.

7. Mark the **1 to Many** radio button at the bottom of the window.

   This specifies that the files are joined in a One to Many relationship. There are many Enrollment records for a class.

8. Press the **Add** button.

   The *Link Fields* window appears. This is where you specify the fields which link the files together.

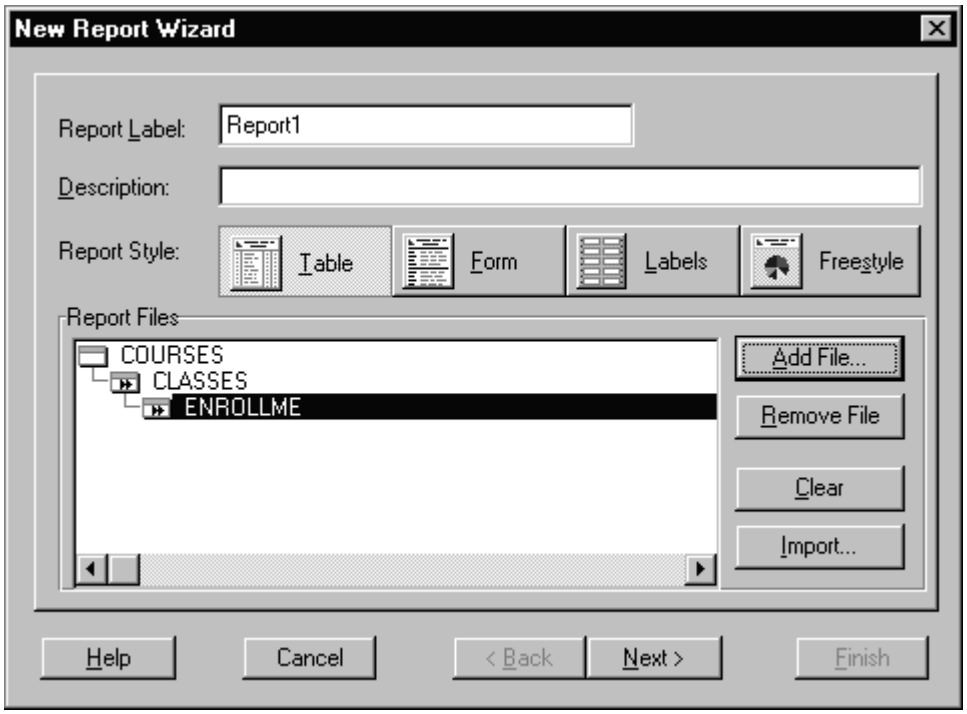
9. Select *CLA:ClassNumber* from the **Link From** drop-down list.

10. Select *ENR:ClassNumber* the **Link To** drop-down list.

This specifies that these fields join the files together.

11. Press the **OK** button on the **Link Fields** window, then press the **OK** button on the *User Defined Relationship* window.

The File Schematic should appear as pictured below.



Adding the Lookup files

1. With *CLASSES* highlighted, press the **Add File** button.

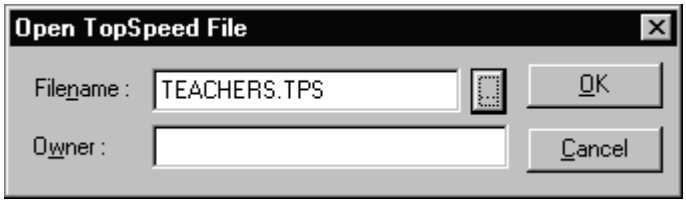
The *Select File* window appears.

2. Press the **Other File...** button.

The *Select File Driver* dialog appears.

3. Select *TOPSPEED* from the **File Driver** drop-down list, then press the **OK** button.

The *Open TopSpeed File* window appears.



4. Press the ellipsis (…) button next to the **Filename** entry box.

   A standard file open dialog appears.

5. Select *TEACHERS.TPS*, then press the **OK** button.

6. Press the **OK** button on the *Open TopSpeed File* window.

   The *User Defined Relationship* window appears. This is where you define the relationship.

7. Mark the **Many to 1** radio button at the bottom of the window.

   This specifies that the files are joined in a Many to One relationship. There is one Teacher for many classes.

8. Press the **Add** button.

   The *Link Fields* window appears. This is where you specify the fields which link the files together.

9. Select *CLA:TeacherNumber* from the **Link From** drop-down list.

10. Select *Tea:Number* the **Link To** drop-down list.

    This specifies that these fields join the files together.

11. Press the **OK** button on the *Link Fields* window, then press the **OK** button on the *User Defined Relationship* window.

Adding the Student Lookup File

1. With *Enrollme* highlighted, press the **Add File** button.

   The Select File window appears.

2. Press the **Other File...** button.

   The *Select File Driver* dialog appears.

3. Select *TOPSPEED* from the **File Driver** drop-down list, then press the **OK** button.

   The *Open TopSpeed File* window appears.

4. Press the ellipsis (…) button next to the **Filename** entry box.

   A standard file open dialog appears.

5. Select *STUDENTS.TPS*, then press the **OK** button.

6. Press the **OK** button on the *Open TopSpeed File* window.

The *User Defined Relationship* window appears. This is where you define the relationship.

7. Mark the **Many to 1** radio button at the bottom of the window.

   This specifies that the files are joined in a Many to One relationship. There is one Teacher for many classes.

8. Press the **Add** button.

   The *Link Fields* window appears. This is where you specify the fields which link the files together.

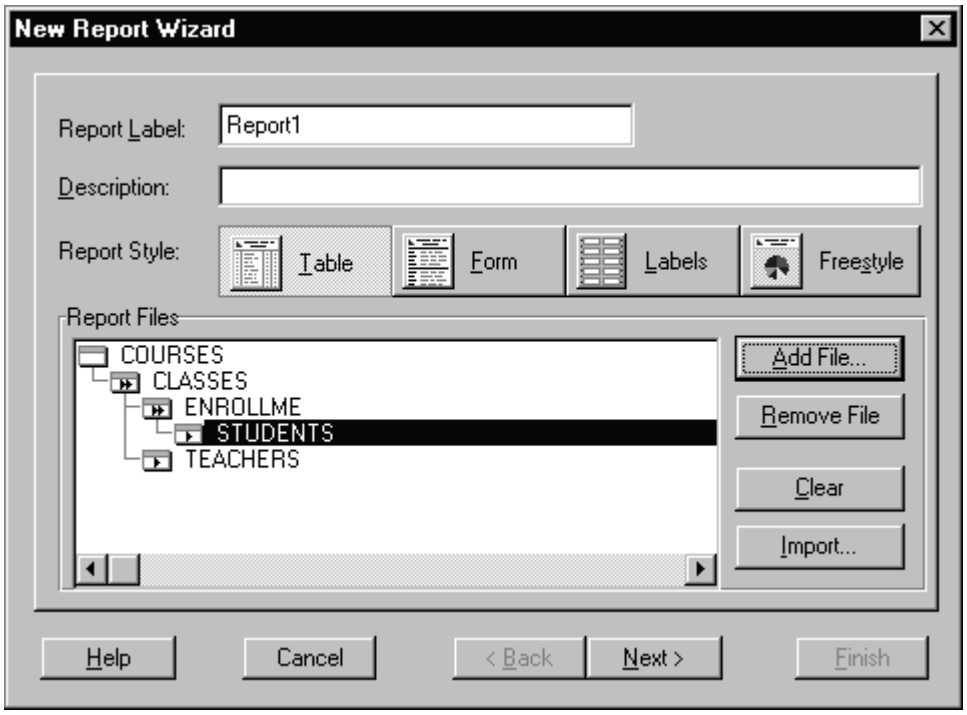
9. Select *ENR:StudentNumber* from the **Link From** drop-down list.

10. Select *STU:Number* the **Link To** drop-down list.

    This specifies that these fields join the files together.

11. Press the **OK** button on the *Link Fields* window, then press the **OK** button on the *User Defined Relationship* window.

    The File Schematic is complete and should appear as pictured below.



12. Press the **Next>** button.

    The *Fields to Populate* window appears.

Populating Fields

In this dialog, you specify the fields to place on the report. You select fields by adding them to the Selected Fields list. The drop-down list above the Available Fields list lets you select the file in the schematic from which you want to populate fields.

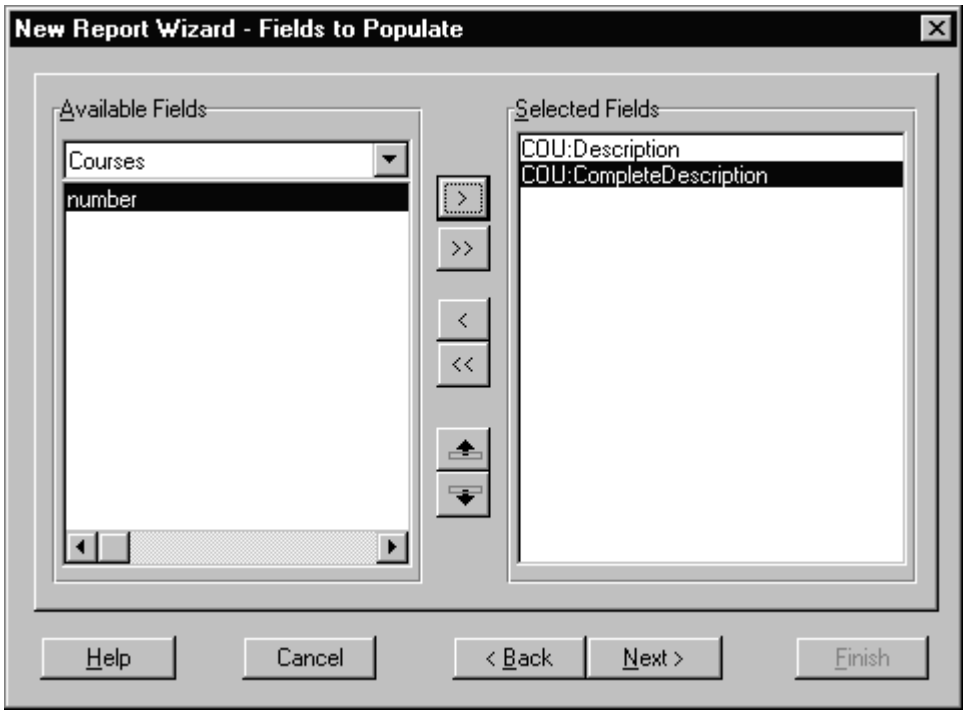
1. Make sure the *Courses* file is selected in the drop-down list.

2. In the **Available Fields** list, highlight *Description*, and press the > button.

   The *COU:Description* field appears in the **Selected Fields** list.

3. In the **Available Fields** list, highlight *CompleteDescription*, then press the > button.

   The *COU:CompleteDescription* field also appears in the **Selected Fields** list.



4. Use the drop-down list above the **Available Fields** list to select the *Classes* file.

5. Press the >> button to populate all the fields in the *Classes* file.

   All the fields in the *Classes* file now appear in the **Selected Fields** list.

6. In the **Selected Fields** list, highlight *CLA:CourseNumber*, then press the < button.

   This removes the *CLA:CourseNumber* from the **Selected Fields** list. Since this is a linking field, you don't need it on the report.

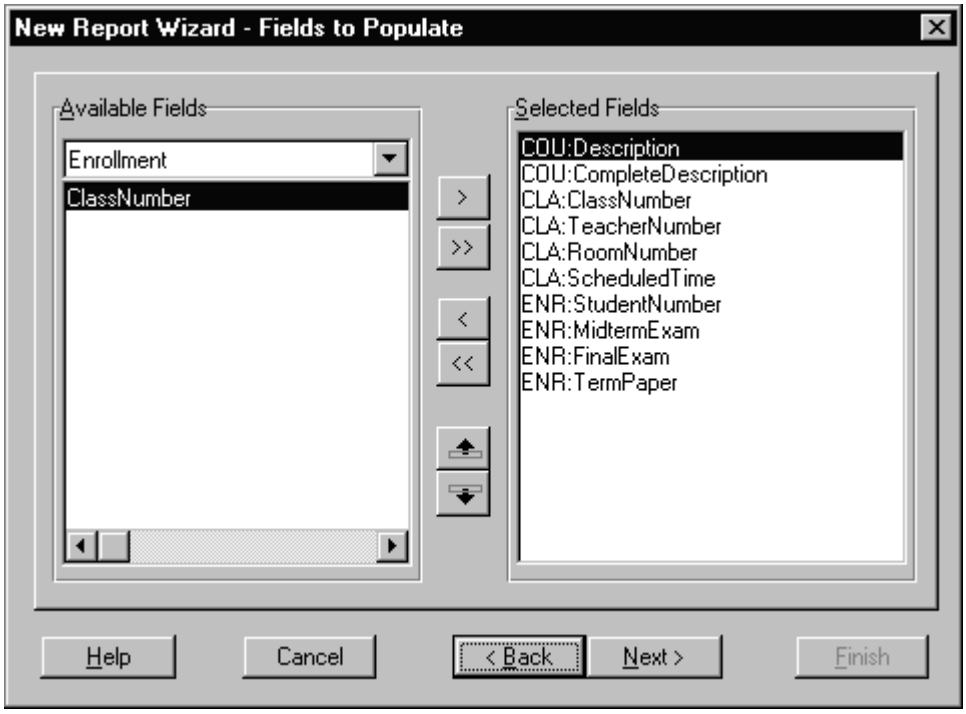
7. Use the drop-down list above the **Available Fields** list to select the *Enrollment* file.

8. Press the >> button to populate all the fields in the *Enrollment* file.

   All the fields in the *Enrollment* file now appear in the **Selected Fields** list.

9. In the **Selected Fields** list, highlight *ENR:ClassNumber*, then press the < button.

   This removes the CLA:ClassNumber from the Selected Fields list. Since this is a linking field, you don't need it on the report. The Fields to Populate window should look like the illustration below:



10. Press the **Next>** button.

    The *Sort Order* window appears.

Specifying the Sort Order and Group Breaks

In this window, you specify the sort order and group breaks for a report. You can sort your report using an existing Key or define a new user-defined sort order. For this report you want to access records in CourseDescription order.

1. Use the **Order By** drop-down list to select *KeyDescription*.

   This key sorts the report by Course's Description.

2. Highlight *COU:Description*, then check the **Group Break** box.

   This will group all related records for a Course together.

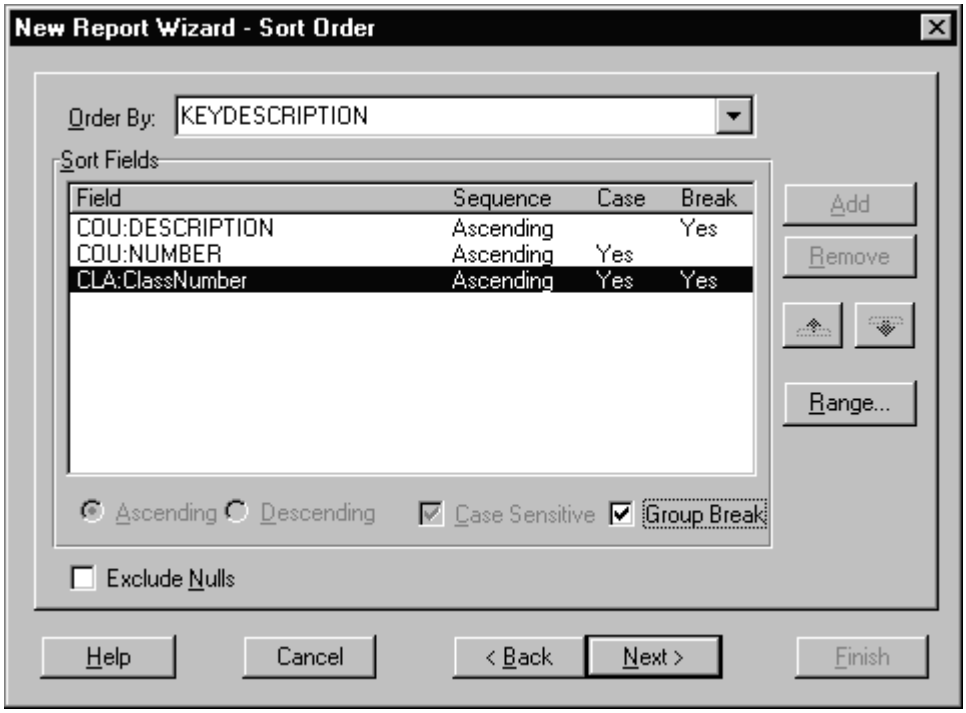
3. Highlight *CLA:ClassNumber*, then check the **Group Break** box.

   This will group all related records for a Class together.

4. Do not check the **Exclude Nulls** box.

   Exclude Nulls specifies that only records which are not zero or blank are considered. Since you want to print Courses and Classes with or without student enrollments, you should include null records.

   The *Sort Order* window should appear as pictured below:



5. Press the **Next>** button.

The Report Layout window appears.

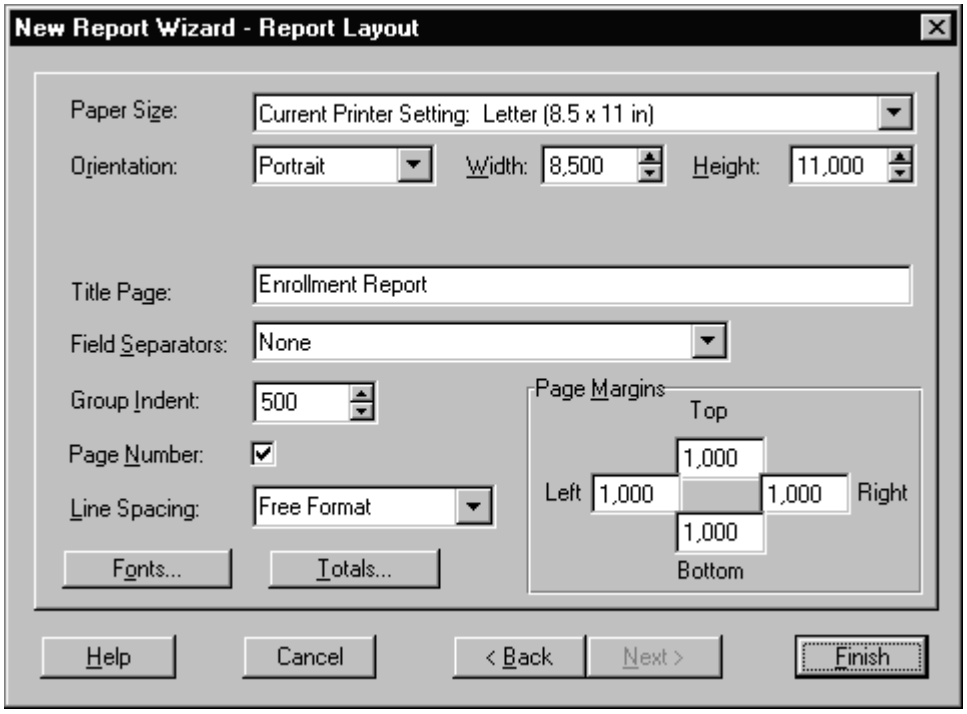
Specify the Report Layout

This window lets you specify a number of options to fine-tune your report's layout. For this tutorial, you will use most of the default settings.

1. Skip the **Paper Size** field.

   This field lets you specify one of a number of common paper sizes. In most cases (and for this tutorial), you will use the *Current Printer Setting*.

2. Make sure the Orientation is set to *Portrait*.



3. Skip the **Width** and **Height** fields.

   These fields allow you to specify custom paper sizes.

4. In the **Title Page** field, type *Enrollment Report*.

   This creates a cover page for your report. If you leave this blank, the wizard creates no title page.

5. Select *Horizontal* from the **Field Separators** drop-down list.

   This will place horizontal lines under each row of fields to make it easier to read.

6. In the **Group Indent** field, specify *500*.

   This specifies the amount of indentation for each group level of the report. For this report, there are two group breaks, so each level is indented 500/1000ths of an inch (i.e., one-half inch).

7. Check the **Page Number** box.

   This adds a page number to the bottom of each page in a Page Footer band.

8. Skip the **Page Margins** section.

   This sets the margins for report. For this tutorial, the default values are adequate.

9. Press the **Finish** button.

   The Report Wizard process the information you supplied, creates the report, and opens the Report Formatter.

At this point, you can press the  button to preview the report. When you are finished previewing, press the  button to return to the Report Formatter.

Conditional Detail Printing

You can use conditional detail bands to print a band only when a specific condition is true. If you previewed the report, you may have noticed blank records printing in the detail band. Adding a condition to the band can suppress those blank details.

In the Detail Band:

1. RIGHT-CLICK on the Detail band, then select **Properties**.

   Make sure you RIGHT-CLICK on an unused portion of the band.

2. In the *Property List*, click on the **Condition** property, then press the ellipsis (...) button.

   The Formula Editor appears.

3. Press the **NOT** button on the Formula Editor keypad.

4. Highlight *System Functions* in the list box on the left, then highlight *EMPTY('filename')* in the list box on the right, then press the **Insert Function** button.

   The EMPTY function determines if a record set is empty. For example, when no enrollment records exist for a class. When you select this function, a *Select File* window appears.

5. Highlight *Enrollment* in the list box, then press the **Select** button.

6. Press the **OK** button.

7. Press the **Close** button on the *Property List*.

Variable Length Memos on Reports

The COU:Description has a MEMO field populated on it. This is a MEMO field in the data file. Memo fields are generally used to store data of varying lengths. One record may contain several lines of text in its Memo, while another may contain a few words. Typically, you will want your report to adjust to the size of the Memo. If it were a fixed size, you would have blank areas on the report.

The Report Wizard automatically created the Group Header Band to handle the varying length of the Memo Field. When you previewed the report, you probably noticed that some course descriptions were long and others were brief. The report printed with no noticeable "gaps."

This is accomplished with a combination of two property settings. First the TEXT control for the Memo field is set to Auto Expand. This specifies that the Text control expands to accommodate all the data in the memo. The second property is on the Group Header Band. It is set to not be a Fixed Height. This allows the band to expand and contract to accommodate the size of the controls within the band.

To examine the properties used to produce this effect:

1. RIGHT-CLICK on *COU:CompleteDescription*, then select **Properties**.

   Notice **Auto Expand** is set to *Yes*.

2. Press the **Close** button.

3. RIGHT-CLICK on the *COU:Description* Group Header Band, then select **Properties**.

   Notice **Fixed Height** is set to *No*.

4. Press the **Close** button.

   Using these two properties, you can create expanding Memos and Bands in any report.

Creating Group Footers with Total Fields

Before total fields can be placed on a report, they must be defined. Total fields are report-specific. This means they are specific to one report and are calculated at runtime. There are two total fields you want for this report: One to count the number of students in a Course, the other to count the total number of students in a Class. These two total fields are similar. The only difference is the point where each resets to zero. One resets after each Class, the other resets after each Course.

Create the Group Footers

1. Choose **Band** ▸ **Group Footer**, then select *COU:Description*.

   This creates a Group Footer band for *COU:Description* (the group break specified on the Course Description).

2. Choose **Band** ▸ **Group Footer**, then select *CLA:ClassNumber*.

   This creates a Group Footer band for *CLA:ClassNumber* (the group break specified on the Class Number).

Create the Total Fields

Next create total fields to count the number of students in each class and the number of students in each Course.

1. If the Control Toolbox is not active, press the  button on the toolbar.

   This displays the Control Toolbox.

2. Press the  button on the Control Toolbox.

   The **Field Total** window appears. This is where you define the total field.

3. Use the drop-down list at the top of the Field List to select the *Enrollment* file.

4. Highlight *StudentNumber* in the **Fields** list box.

   This specifies that the Student Number from each Enrollment record is used for the calculation.

5. In the **Total Type** group, mark the **Count** radio button.

   This specifies that the total field will count Student Numbers.

6. In the **Picture** entry box, type *@N5* (or press the ellipsis button to use the Picture Editor).

   This specifies the total field will print in a four-digit format, with a comma separating the thousands column (e.g., 9,999).

7. Use the drop-down list next to the **Tally** field to select *ALL*.

   The **Tally** on *All* specifies that every occurrence of *ENR:StudentNumber* is considered.

8. Use the drop-down list next to the **Reset On** field to select *COU:Description*.

   This specifies that the total field resets to zero when the group break for a new Course occurs.

9. Press the **OK** button to return to the formatter.

   Notice the mouse cursor has "turned into" an ABC cursor. That indicates it is "holding" a control to place on the report. When you click on any band of the report, the control is placed at that spot.

10. Inside the *COU:Description* footer band, CLICK on the location where you want to place the total field.

Create another total field to count the students in each class

1. Press the  button on the Control Toolbox.

   The **Field Total** window appears allowing you to define the total field.

2. Use the drop-down list at the top of the Field List to select the *Enrollment* file.

3. Highlight *StudentNumber* in the **Fields** list box.

   This specifies that the Student Number from each Enrollment record is used for the calculation.

4. In the **Total Type** group, mark the **Count** radio button.

   This specifies the total field will count Student Numbers.

5. In the **Picture** entry box, type *@N5* (or press the ellipsis button to use the Picture Editor).

   This specifies the total field will print in a four-digit format, with a comma separating the thousands column (e.g., 9,999).

6. Use the drop-down list next to the **Tally** field to select *ALL*.

   The **Tally** on *All* specifies that every occurrence of *ENR:StudentNumber* is considered.

7. Use the drop-down list next to the **Reset On** field to select *CLA:ClassNumber* band.

   This specifies that the total field resets to zero when the group break for a new Course occurs.

8. Press the **OK** button to return to the formatter.

   Again the mouse cursor has “turned into” an ABC cursor, indicating it is “holding” a control to place on the report. When you click on any band of the report, the control is placed at that spot.

9. Inside the *CLA:ClassNumber* footer band, click on the location where you want to place the total field.

Populating String Controls to Identify the Totals

You’ll want to identify these total fields by printing some text. A simple String control with some text will suffice.

1. Press the 

button on the Control Toolbox.

   Again the mouse cursor has “turned into” an ABC cursor, indicating it is “holding” a control to place on the report. When you click on any band of the report, the control is placed at that spot.

2. Inside the *CLA:ClassNumber* footer band, CLICK on the location (to the right of the total field) where you want to place the total field.

   When you place the control, it is in Edit Mode. In Edit Mode you can edit the text directly. If you later want to edit the text, you can DOUBLE-CLICK on the control.

3. Inside the String Control, type *Students Enrolled*, then press ENTER.

   The String control you want for the other total field is exactly the same, so you can use ReportWriter’s copy and paste feature.

4. CLICK on the *Students Enrolled* String Control, then choose **Edit** ▸ **Copy** (or press CTRL+C).

5. CLICK on the *COU:Description* footer band, then choose **Edit** ▸ **Paste** (or press CTRL+V).

6. Position the control (to the right of the total field) by DRAGGING it with the mouse or pressing the cursor arrow keys.

Adding Lookup Fields

Next you will add some fields from "lookup" files to provide more understandable data. For example, for each Class you are printing a Teacher Number which uniquely identifies a teacher. But it will be more understandable to print the teachers' names on the report.

These data fields are populated in the same manner as any other field.

1. CLICK on the  on the toolbar to open the *Field Selection List* (or choose **View** ▸ **Field Selection List**).

   The *Field Selection List* appears. This "floating" tool lets you quickly place data fields on your report.

2. Select *Teachers* from the drop-down list on the *Field Selection List*.

   The Fields in the *Teacher* file appear on the list.

3. CLICK on *LastName* in the list, then click on the location of the report where you want to place the field.

   You should place the *LastName* field in the *CLA:ClassNumber* Group Header band.

4. CLICK on *FirstName* in the list, then click on the location of the report where you want to place the field.

   You should also place the *FirstName* field in the *CLA:ClassNumber* Group Header band beneath the *LastName* field.

5. Select *Students* from the drop-down list on the Field Selection List.

   The Fields in the *Students* file appear on the list.

6. CLICK on *LastName* in the list, then CLICK on the location of the report where you want to place the field.

   You should place the *LastName* field in the detail band to the left of the *ENR:StudentNumber* field.

   Your report now prints the desired information. If you want to, preview the report to see the results.

Adding a Computed Field

There is only one item left from the original plan for the report—printing each student's final grade. The Final Grade is a weighted calculation based on each individual grade (Midterm Exam, Final Exam, and Term Paper). The developer who designed this database did not store the grade in the student file because it is easily calculated using the data already stored in the file. To store the final grade in the student file would waste space (data redundancy).

This college has a standard formula for computing a student's final grade—the Mid-Term Exam is worth 25%, the Final Exam is worth 25%, and the Term Paper is worth 50%.

To print the final grade, create a Computed field using this formula:

```
(ENR:MidTermExam * 0.25) + (ENR:FinalExam * 0.25) + (ENR:TermPaper * 0.5)
```

Create the Computed Field

Next create the Computed field to calculate the final grade for each student.

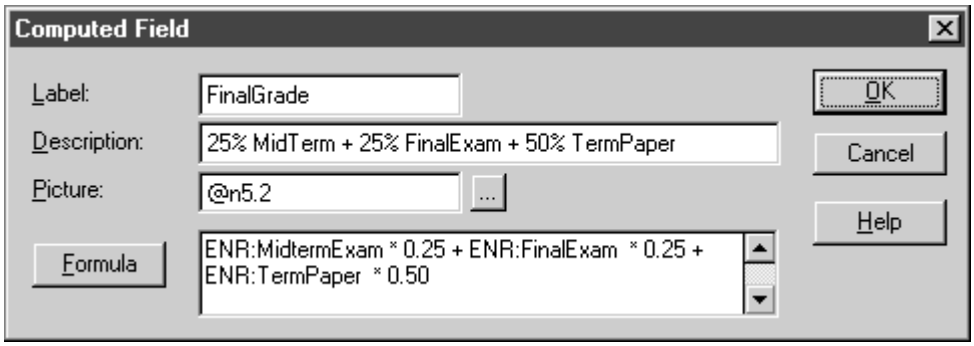
1. If the Control Toolbox is not active, press the  button on the toolbar. This displays the Control Toolbox.

2. Press the  button on the Control Toolbox.

   The *Insert Data Field* window appears. This is the same as the report's File Schematic.

3. Highlight Computed Fields in the list box on the left, then press the **Add Computed Field** button. The *Computed Field* window appears.

4. Create the Computed Field in the same manner as before, then press the **OK** button. The completed *Computed Field* window should look like the illustration below:



5. In the *Insert Data Field* window, highlight *Computed Fields* in the list box on the left, then highlight *FinalGrade* in the list box on the right, then press the **Select** button.

6. CLICK on the Detail band at the location of the report where you want to place the field (to the right of the *ENR:TermPaper* field).

7. Place a String control in the *CLA:ClassNumber* Group Header Band to serve as a column header for the Final Grade.

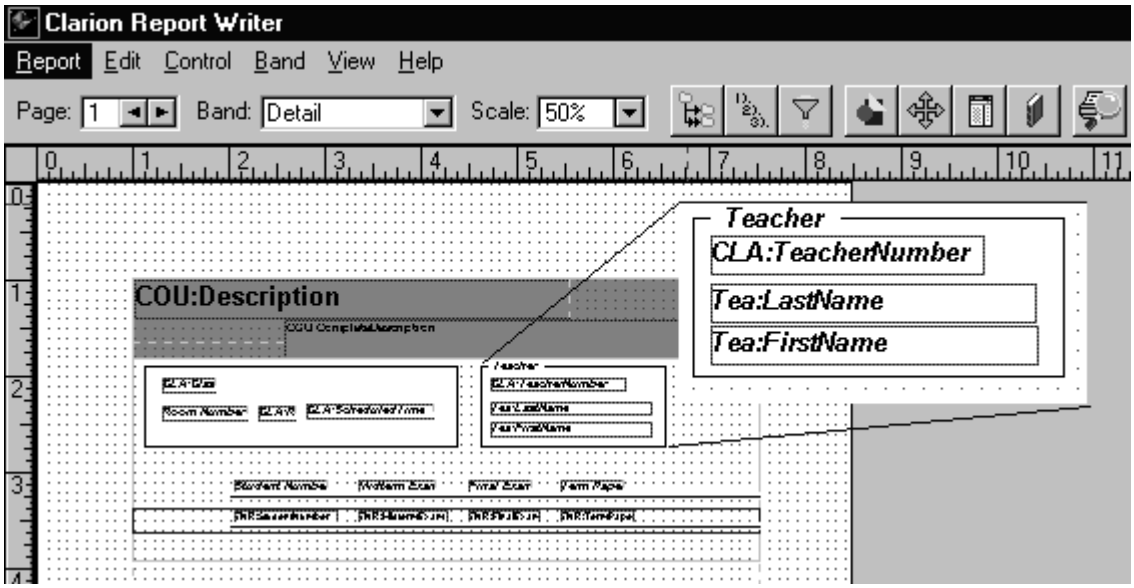
Cosmetic Enhancements for the Report

The report is fully functional now, but you can improve it with little effort using some of ReportWriter's cosmetic features.

Grouping Controls using a Group Box

The GROUP control lets you reference a group of controls as one entity. When you set properties for the GROUP, all controls within that group inherit the property setting. You can override the inherited setting for an individual control by explicitly setting the same property for the control.

Another use for a Group box is to display a box around a group of controls and optionally provide text to identify the group. An example is illustrated below:



For this report, putting fields within a group box can also eliminate the need for prompts next to the Teacher fields. The Group Box text identifies the data.

Create a Group Box for the Teacher Information

1. If the Control Toolbox is not active, press the button on the toolbar.

   This displays the Control Toolbox.

2. Press the button on the Control Toolbox.

3. CLICK on the *CLA:ClassNumber* Group Header band at the location where you want to place the Group Box (somewhere on the right side).

4. CLICK and DRAG the *Tea:LastName*, *Tea:FirstName*, and *CLA:TeacherNumber* fields into the group box.

5. Resize the group box as needed by CLICKING and DRAGGING its handles.

6. Delete the *Teacher Number:* prompt (String control).
7. RIGHT-CLICK on the group box, and select **Properties**.
8. In the Text property, type *Teacher*, then close the *Property List*.

   The text now appears at the top of the Group Box.

Create a Group Box for the Class Information

1. Press the  button on the Control Toolbox.
2. CLICK on the *CLA:ClassNumber* Group Header band at the location where you want to place the Group Box (somewhere on the left side).
3. CLICK and drag the *CLA:ClassNumber*, *CLA:RoomNumber*, and *CLA:ScheduledTime* fields into the group box.
4. Resize the group box as needed by CLICKING and DRAGGING its handles.
5. Delete the associated prompts (String controls).
6. RIGHT-CLICK on the group box, and select **Properties**.
7. In the Text property, type *Class*, then close the *Property List*.

   The text now appears at the top of the Group Box.

Using Different Fonts

Different font properties can make elements of your report stand out. ReportWriter lets you specify the font's Typeface, Size, Color, and Style. Use any combination you desire to create the report you want.

For this report, you want to make the font on a Course Description larger and Bold. That will make a better Group Header and distinguish it from the other elements of the report.

In the COU:Description Header Band:

1. Delete the String Control that says *Description:* and the String Control that says *Complete Description:*

   You won't need the prompts to identify the fields.
2. Move the *COU:Description* field all the way to the left by CLICKING and DRAGGING it with the mouse.
3. RIGHT-CLICK on the *COU:Description* field, then choose **Font**.

   The *Choose Font* dialog appears. This dialog lets you modify all of the Font Properties at the same time.
4. Select *Arial* as the Font, *Bold* as the Font Style, and *20* as the Size, then press the **OK** button.

5. Move the *COU:CompleteDescription* field down (so it is not obscured) by CLICKING and DRAGGING it with the mouse.

Using Background colors

You can also use background colors to enhance elements on your report. If you have a color printer, you can print the background colors. On a non-color printer, the colors are converted to greyscale by the printer's driver. For this report, a grey background would be nice on the *COU:Description* Header Band.

In the COU:Description Header Band:

1. RIGHT-CLICK on the *COU:Description* Header band, then select **Properties**.

   Make sure you RIGHT-CLICK on an unused portion of the band.

2. In the *Property List*, press the ellipsis (...) button next to **Background**.

   A standard windows color dialog appears.

3. CLICK on the grey-colored box below **Basic Colors**, then press the **OK** button.

4. Press the **Close** button on the *Property List*.

   The band now has a grey background color. Controls on a band take on the band's properties unless a property is explicitly set for a control. Therefore, the controls on this band take on the background color of the band.

Controlling Pagination

There are four properties which control the manner in which report elements span pages (pagination). Using these pagination options, you can create a report that prints in an organized manner and ensures that every page is understandable.

Page Before Specifies to print the structure on a new page. To reset the page number to a value you specify, type it in the **Page Before** field. To force a Page Before without resetting the page number, specify -1.

Page After Specifies to print the structure, then force a new page. To reset the page number to a value you specify, type it in the **Page After** field. To force a Page After without resetting the page number, specify -1.

With Prior Prevents 'orphan' elements in a printout. An 'orphaned' print element is one which prints on a following page, separated from its related items. The value specifies the number of preceding elements to print—a value of "1," for example, specifies that the previous element must print on the same page. When placing subtotals or totals in a band, use the WITHPRIOR attribute to insure they print with at least one member of the column above it when a page break occurs.

With Next Prevent 'widow' elements in a printout. A 'widowed' print element is one which prints, but then is separated from the succeeding elements by a page break. The value specifies the number of succeeding elements to print—a value of '1,' for examples, specifies that the next element must print on the same page, else page overflow puts them both on the next.

For this report, use a combination of these properties to control the report's pagination. This time use the floating Property list to set the properties.

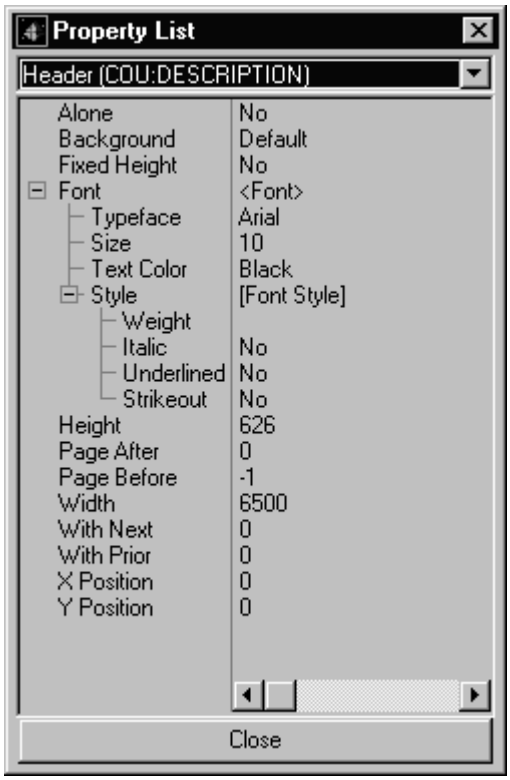
1. CLICK on the  button.

   The floating *Property List* appears. This toolbox allows you to set properties of any currently selected element of your report.

2. CLICK on the COU:Description Header band.

3. In the **Page Before** entry box in the Property List, type *-1*.

   This forces a page break before each group header, ensuring that each new course starts at the top of a page. By specifying -1, page numbers are not reset and continue to increment throughout the report.



4. CLICK on the *COU:Description* Footer band.

5. In the With Prior entry box in the Property List, type *1*.

   This ensures that each course footer prints with at least one prior element of the report (the CLA:ClassNumber footer).

6. CLICK on the CLA:ClassNumber Footer band.

7. In the With Prior entry box in the Property List, type *1*.

   This ensures that each Class footer prints with at least one prior element of the report (the detail band).

The report is complete. Preview the report, then close it.

Congratulations! You have completed this tutorial.

Summary

- When creating relational reports, a little planning in advance can save a lot of time.
- When you import a Clarion Data Dictionary, ReportWriter imports file relationships defined in the dictionary or you can create user-defined relations.
- Link Fields define the relationship between files.
- When defining the file schematic for your report, you add related files to the currently highlighted file.
- The **Sort Order** dialog lets you specify the order in which records print and set group breaks to group records together.
- To exclude "empty" records, use the **Exclude Nulls** option.
- The Report Wizard automatically handles the varying length of Memo fields by setting the **Auto Expand** property on the Text control and not setting the **Fixed** property on the band.
- Group Footers print after all Detail bands for a group print.
- Relations allow you to lookup values from a parent file to print on a report.
- Group Boxes allow you to group controls together.
- Background colors and Fonts allow you to highlight elements of your report.
- Control pagination using the **Page Before**, **Page After**, **With Next**, and **With Prior** properties.

7 - USING REPORT LIBRARIES

Report Libraries–An Overview

A Report Library is a collection of reports based on a common data set. There are several ways to define this data set.

- Create the library based on a Clarion Data Dictionary (either .DCT or .TXD format). A Data Dictionary provides more information than a data file. The data dictionary provides all default report control picture formatting, prompts, keys, and relations between files. A Library based on a dictionary also has the ability to provide automatic synchronization when the dictionary is modified.
- Create the library based on a Clarion ReportWriter for Windows Report Library (.TXR). This provides all of the data file information contained in the original Report Library.
- Create the library based on a data file. This provides all of the information stored in the data file.

The *Report Library* dialog lets you:

- Create, Edit, or Delete Reports
- Copy Reports (within the Library)
- Import Reports from another Report Library
- Import data file definitions from data files, Clarion Data Dictionaries, or other Report Libraries
- Preview or Run Reports.

Creating a New Report Library

1. Press the  button on the toolbar (or choose **File** ▸ **New**).

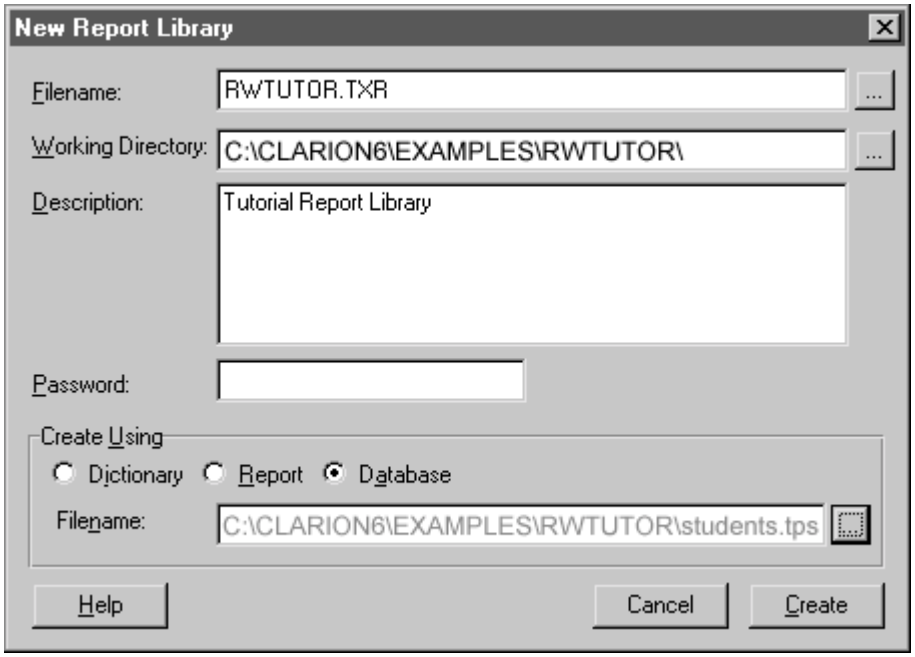
   If a Report Library is open, you are prompted to save the open library. If you want to save it, press the **OK** button.

   The *New Report Library* window appears. This is where you specify the basic information for the Report Library and give it a name.

2. In the **Filename** entry box, type a name for the Report Library (without the extension). Optionally, you can press the ellipsis (...) button to select a different directory.

   ReportWriter automatically adds the .TXR file extension.

3. In the **Working Directory** field, type the path of the working directory or press the ellipsis (...) button to select it from a standard file dialog.



4. Optionally, in the **Description** text box, type a description for the Report Library.

   Although this is not required, a description helps you identify the report library.

5. Type a password in the **Password** field if you want password protection for your library.

   Password protection is optional and prevents unauthorized access to this set of reports. No one can modify this set of reports without the password. If desired, you can use the same data source to make other sets of reports that allow access to other users.

   Note:

   Using a password encrypts the Report Library File (.TXR) , preventing anyone from viewing it in a text editor. If you want encryption without a password, use a single asterisk (*) as the password. This prevents casual viewing in a text editor without requiring a password.

6. Select the source from which to create the Report Library in the **Create Using** section. The choices are:

   Dictionary A Clarion Data Dictionary (.DCT or .TXD).

   Report Another Report Library (.TXR)

   Database An existing database file.

7. In the **Filename** entry box, specify the file from which to create the Report Library by using the ellipsis (...) button to select it from a standard File Open dialog.

8. Press the **Create** button.

9. Create reports for the library (see the *Creating Reports* chapter).

Editing Report Library Properties

You can modify the *Report Library Properties* window after it has been created. This lets you add or modify the Library after you have created it.

Each time you create a new Report Library file, the *Report Library Properties* window automatically appears before the creating the file. You can also access the *Report Library Properties* window for an existing file from the Report Library list.

1. Press the **Properties** button (or choose **Report ▸ Library Properties**).
2. Modify the properties as desired.
3. Press the **OK** button.

Report Editing Modes

A report library can be edited in one of two modes: a) Developer Mode (which is the default) and b) End User Mode.

In End User Mode, some fields and files can be hidden in listbox selections. This allows the developer to hide unneeded or sensitive fields from the target end-user who uses ReportWriter to design new reports.

Fields and files may be hidden by checking the respective “Hidden from end user” checkboxes on the Field Properties Dialog (see above). Remember, modifying Field Properties can only be done in developer mode.

To set a Report Library Mode from the Report Library window:

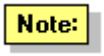
1. Choose **Report ▸ Library Properties**.

   The *Report Library Properties* window appears.
2. Check or (clear) the **End-User** box.

A Report Library is set to End User mode by checking End-User Mode box on the Report Library Properties dialog. To set a Report Library to Developer Mode, clear the box.

If the report has a Developer Password, then this password must be entered when changing from end-user mode back to developer mode. This allows the End User Mode to be ‘locked’ on by the developer.

If there is no main password but there is a developer password then the TXR is encrypted with a standard password to prevent alteration.



The normal and developer passwords can only be changed in developer mode.

Developer/End-User mode and the 'descriptive' modes are stored on a report library basis within the TXR allowing them to be set-up by the developer before distributing the report library to the end user.

Accessing a Single record File

You can use a single record (control) file which is not related to any other files by adding the file as a user-defined relation and not specifying any link fields. This allows you to populate fields from that file to print data from the first record in the file.

1. CLICK on the  button or choose **Report** ▸ **File Schematic**.

   The *File Schematic Definition* window appears. This contains a list box which displays all the files used in a report. The list box is divided in two sections: *Files* and *Other Data*.

2. With the primary file highlighted, press the **Add File** button.

   The *Select File* window appears. This window has 2 sections: Related Files and User-defined Related Files. The first section displays the files that have defined relations to the Primary file in the Database Dictionary. The second lists all the remaining files in the Report Library.

3. Highlight the single record file in the *User Defined Related Files* section and press the Select button.

   The *User Defined Relationship Properties* window appears.

4. Press the **OK** button (without specifying any link fields).

   A warning appears to let you know that you have not specified any links.

5. The link appears in the **Link Fields** list box in the *User Defined Relationship Properties* window.

   The file schematic now displays the files and the new "user-defined relationship." Populating any fields from this file will get the values from the first record in that file.

Adding Reports to the Report Library

The *Report Library* lists all the reports in the Report Library file.

When you create a Library, a dialog appears asking if you want to create a report. Press the **Yes** button to continue and create a new report.

To create a new report:

1. Press the **New** button.

   The Report Wizard appears.

2. Design the report using one of the Report Wizards and the Report Formatter (see the *Creating Reports* chapter).

Managing Reports in a Report Library

Cut, Copy, and Paste

When you want to create a report that is similar to an existing one, you can copy the existing report, then modify the new one. This time saving feature simplifies the task of creating reports.

1. Highlight the report to copy in the **Report Library** list.
2. Choose **Report** ▸ **Copy** (or press CTRL+C).
3. Choose **Report** ▸ **Paste** (or press CTRL+V).

   ReportWriter copies the report and provides a new name. All Reports within a Report Library must have unique names.
4. Optionally, modify the name and description by pressing the **Properties** button.
5. Highlight the new report, then press the **Edit** button.
6. Make the desired modifications.

You can also cut, copy, and paste between Report Libraries. Open a library and cut or copy a report, close the library and open another. Then, paste the report into the second library. Keep in mind that the file definitions must already be present in the target library.

Importing File Definitions and Reports

ReportWriter needs file definitions for every file for which a report will be created. There are several different ways to import file definitions. You can also import file definitions at report design time using the file schematic. This is explained in the Creating Reports chapter.

Importing File Definitions from a Clarion Data Dictionary

If the application which maintains your database was created with Clarion, it is probably based on a Data Dictionary. The Data Dictionary is a repository for information relating to the data files, their keys, and their relationships. If you have a data dictionary (.DCT or .TXD) file, this is the best source from which to create a Report Library. A Data Dictionary contains more information than the data file. It also contains the relations between files.

In order to import a Data Dictionary(.DCT), you must have Clarion installed and it must be running. You do not need Clarion to import a textual representation of the Data Dictionary (.TXD).

ReportWriter detects changes to Dictionary or Report Library files used as the basis for a Report Library and lets you incorporate those changes.

If you have more than one user creating reports with ReportWriter and those users do not have Clarion, you should base your reports on a .TXD. This allows users to update any dictionary changes without requiring a copy of Clarion. If the dictionary changes, you merely provide the end users with a new version of the .TXD or .TXR.

Contact the Clarion Programmer who wrote your application to request a .TXD version of your Data Dictionary or a Report Library (.TXR) containing the Data Dictionary.

Importing from a Report Library

You can import reports from an existing Report Library. This lets you create library files that are subsets of other libraries and import other reports into them as needed. When importing a Report Library, you also import all the file definitions and any defined relations between the data files.

The ability to import reports also lets you create a "shell" report file with "template" reports that other users can import and modify. These reports can have File Schematics already specified, eliminating the need to specify schematics when designing a report.

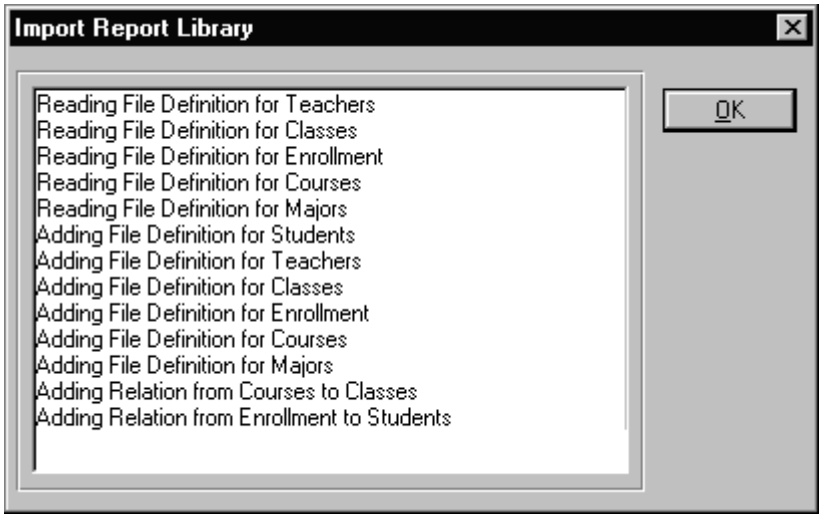
To import a report:

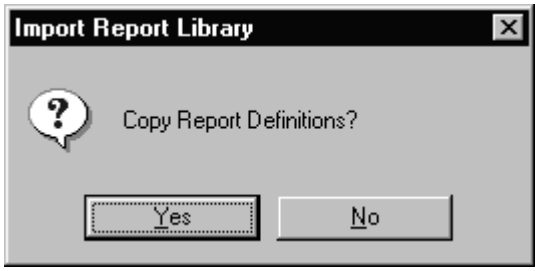
1. Choose **File** ▸ **Import** ▸ **Report**.

   The *Select Report* dialog appears. This is where you specify the Report Library that contains the report to be imported.

2. Highlight the **Report Library File** in the standard file dialog, then press the **OK** button.

   ReportWriter imports the file definitions and relations, then asks if you want to import the reports from that library.





3. If you want to import the Reports, press the **Yes** button. If you don't want to import the reports, press the **No** button. Highlight the **Report Library File** in the standard file dialog, then press the **OK** button.

   If you only want some of the reports, import the reports then delete the ones you don't want.

4. Press the **OK** button to close the **Import Report Library** window.

5. Create or Modify reports as desired.

Deleting a Report from the Report Worksheet

1. Highlight the report to delete in the *Report Library* dialog.

2. Press the **Delete** button (or press DELETE).

3. Press the **Yes** button in the confirmation window.

Saving a Report Library File

1. Press the  button on the toolbar (or choose **File** ▸ **Save**).

   This saves your work in progress. You also have the opportunity to save your work when you close a Report Library, or you can set the ReportWriter option to auto-save when closing (See *Configuring ReportWriter Options*).

Saving a Report Definition File to a Different Filename

The **Save As** command lets you make a copy of the current Report Library and work on the copy.

1. Choose **File** ▸ **Save As**.

   A standard file dialog appears.

2. In the **Filename** field, type a name (without the extension).

3. Press the **OK** button.

   ReportWriter saves the Report Library using the new name. The new Library is now active.

4. If desired, edit **Report Library Properties**.

5. Edit the reports (see the *Creating Reports* chapter).

Specifying the Report Display Order

You can change the order in which reports are listed in the Report Library list by moving reports up or down.

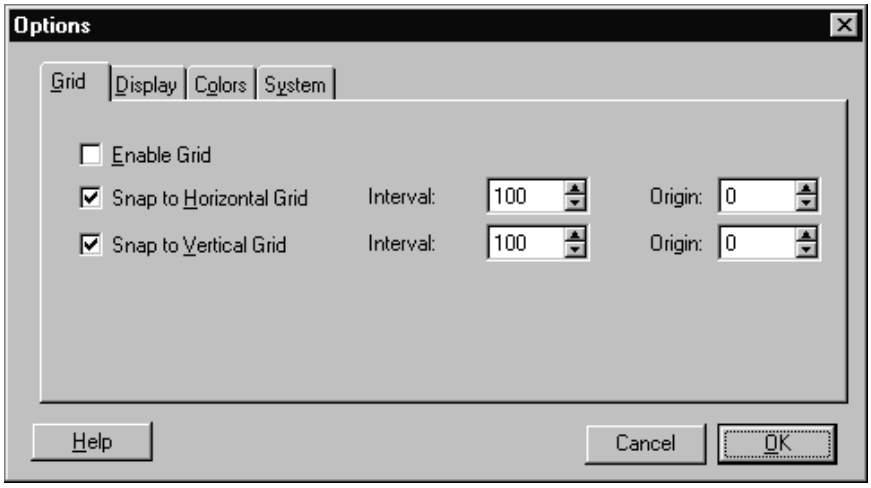
1. Highlight the report to move.
2. Press the  or the  button.

Configuring ReportWriter Options

ReportWriter lets you specify your personal preferences for the Report Formatter's appearance, displays, and system behavior. These settings are saved and remain in effect until you modify them again.

Option settings are global, that is they are not specific to a single report. However, you can change them while editing a report.

To modify your options from the Report Library or the Report Formatter choose **Report ▸ Options**. The Options dialog appears displaying four tabs. Each of these tabs lets you modify specific Report Formatter options.



Grid

Enable Grid	Enables grid markers. These provide visual marks to assist you in control alignment.
Snap to Horizontal	Specifies "snap to" behavior for control placement and movement. If enabled, controls adhere to the horizontal markers on the grid. If disabled, controls can freely move along the horizontal axis.
Interval	Specifies the space between the grid markers for the horizontal axis.
Origin	Specifies the point of origination for the grid along the horizontal axis.
Snap to Vertical	Specifies "snap to" behavior for control placement and movement. If enabled, controls adhere to the vertical markers on the grid. If disabled, controls can freely move along the vertical axis.
Interval	Specifies the space between the grid markers for the vertical axis.
Origin	Specifies the point of origination for the grid along the vertical axis.

Display

Measurement Units
Specifies the unit of measurement for report layout. The choices are : 1/1000th of an inch, 1/10th of a millimeter, or points (1/72 of an inch).

Show Obscured Controls
Allows controls which are covered by other controls to show through.

Frame Inactive Controls
Displays a box around controls which are not currently selected.

Frame Inactive Bands
Displays a box around bands which are not currently selected.

Field Display
Lets you specify what displays in data fields on the report. The choices are:

Field Name Displays the name of the field.

Field Picture Displays field picture tokens.

Example Displays any text you place in the Example entry box.

File Description Only
When checked, files display as their description only rather than as 'label' or 'label - description.' If there is no description, then the label displays. See Editing Descriptions.

Field Description Only
When checked, fields display as their description only rather than as 'label' or 'label - description'. If there is no description, then the label displays.

Prefix Field With File
When checked, the file is prepended to field names wherever the file cannot be contextually determined (this is the default mode). If descriptive modes are being used and all field names are unique then this option may be unchecked in order to minimize the name length and reduce display clutter.

Colors

Control Frame
Lets you change the color of the frame around controls. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Active Frame
Lets you change the color of the frame around the currently active element of the report. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Multiple Selection
Lets you change the color of the frame around multiple selected controls. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Current Band
Lets you change the color of the frame around the currently active band. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Other Band

Lets you change the color of the frame around the inactive bands. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Resize Handle

Lets you change the color of the resize handle on the currently active element of the report. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Obscured Control

Lets you change the color of obscured controls. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Vertical Guide

Lets you change the color of the Vertical Guides. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

Horizontal Guide

Lets you change the color of the Horizontal Guides. Press the ellipsis (...) button to select a color from a standard Windows Color Dialog.

System

Limit Preview to

Specifies the number of pages displayed in Preview Mode. To view all pages in set this to zero (0).

Open Last Library on Startup

Specifies that the last Report Library you worked on will reopen when you start ReportWriter.

Make Backup Files

Specifies to make backup copies of Report Libraries while editing. If enabled, a backup copy is save using a .BKR extension.

Autosave on Close

Specifies to save Report Libraries automatically when closing.

New Report Wizard

Lets you specify the settings the Report Wizards use when creating new reports.

Minimum Column Gap

The minimum size of the space between columns.

Maximum Column Gap

The maximum size of the space between columns.

Minimum Prompt Gap

The minimum size of the space between a field and its prompt.

Keyboard Move Increment

Lets you specify the distance a selected item moves each time you move an item using an arrow key on the keyboard.

Guide Snap Range

Lets you specify the proximity to a grid point which causes the control to snap to the grid.

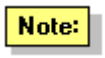
Mouse Movement Threshold

Lets you specify the distance a control must move before it redisplays. This reduces "bounce" while moving items with a mouse.

Editing Descriptions

The Descriptions originate in the data dictionary (.DCT or .TXD) and are stored in the Report Library (.TXR) if it is created from a dictionary. In addition, these descriptions can be modified in a .TXR. The .TXR containing modified descriptions can be used as the source to create other Report Library files. In this manner, any modifications will cascade to "child" .TXR files upon creation. Any subsequent description changes made in a source dictionary will cascade to a Report Library which was created using the dictionary as its source. Report Libraries created from other Report Libraries do not inherit and subsequent changes.

For example, let's say you had a dictionary where the field labels were ambiguous and did not wish to change the labels for some reason. You could edit the DCT and provide good descriptions. Next you would export to TXD, then create a base TXR from that TXD. That TXD would be deployed to Report Writer users to be used as the source from which to create Report Libraries. At some later date, if a description is changed in the source dictionary, a new TXD would be deployed and the automatic synchronization will cascade the new descriptions into the "child" TXR files.



To edit field descriptions, the Report Library must be opened in Developer Mode. In end User mode, the edit button is disabled.

To Edit a Field's Description:

1. Select the field to modify in the file schematic dialog and press the **Edit** button.

   The *Field Properties* dialog appears, allowing you to change the file and field textual descriptions.

An additional feature is the distinction of 'short' and 'full' display descriptions. These are set by using a description of the form: 'shortdesc|fulldesc', i.e., the short form is separated from the full description by the vertical bar (|) character.

Short descriptions are used in field names and full descriptions are used in listboxes where a more verbose description is useful.

For example you if you had a file description 'Office|Office Names and Addresses' and a field 'HomeAddr|Home Address' then in the file schematic you would see:

```
Office Names and Addresses
       Home Address
```

but on the screen the field populated will be displayed as 'Office:HomeAddr' (or just 'HomeAddr' if the Prefix Field With File option is not checked).

If there is no '|' in the description then the short form is the same as the full form.

The Field Properties dialog also allows you to modify the default picture used when populating a field. This is especially useful when creating reports from imported data files. For example, a dBase file might store a date field in mm/dd/yyyy format and you may want the default picture to be mm/dd/yy. By allowing the user to modify the default picture in the Report Library, the time to create subsequent reports is reduced.

Hiding Fields from End Users

At times it is necessary to allow certain users to create and run reports on files that contain sensitive data. The fields containing sensitive data can be hidden from these users to maintain security. For example, you may want to allow a set of users to create reports on the Employee file, but don't want them to have access to salary information. You can also hide files from the end user. Another benefit of hiding fields is to provide a shorter list of fields to the end user. If you have any fields that the end user does not need, hiding them reduces potential clutter.

To hide a Field or File:

1. Select the field to modify in the file schematic dialog and press the **Edit** button.

   The *Field Properties* dialog appears, allowing you to change the file and field textual descriptions.

2. Check the **Hidden from End User** box.

8 - CREATING REPORTS

Using the Report Wizards

Clarion ReportWriter has built-in wizards to create reports quickly. This enables you to provide a little information and have a fully functional report created. After you create a report using a wizard, you can modify it with the Report Formatter. This provides both rapid report prototyping and flexibility.

There are four Report Wizards:

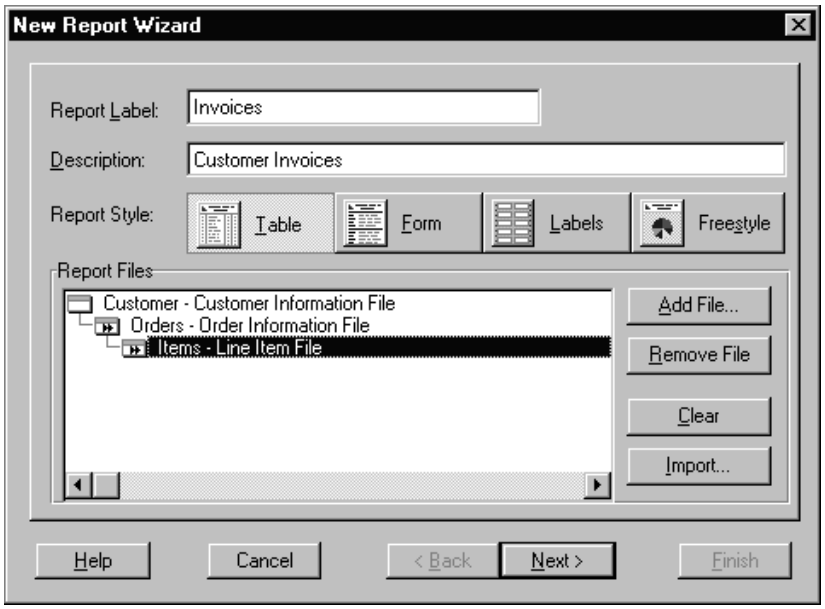
Table	Creates a columnar report.
Form	Creates a form-based report.
Label	Creates a multi-up column report, most commonly used to print labels.
Freeform	Creates a freeform report that you can modify in any manner. This wizard populates no fields, providing you with a “blank slate” to create reports.

To create a report using a Report Wizard:

1. From the Report Library, press the **New** button.

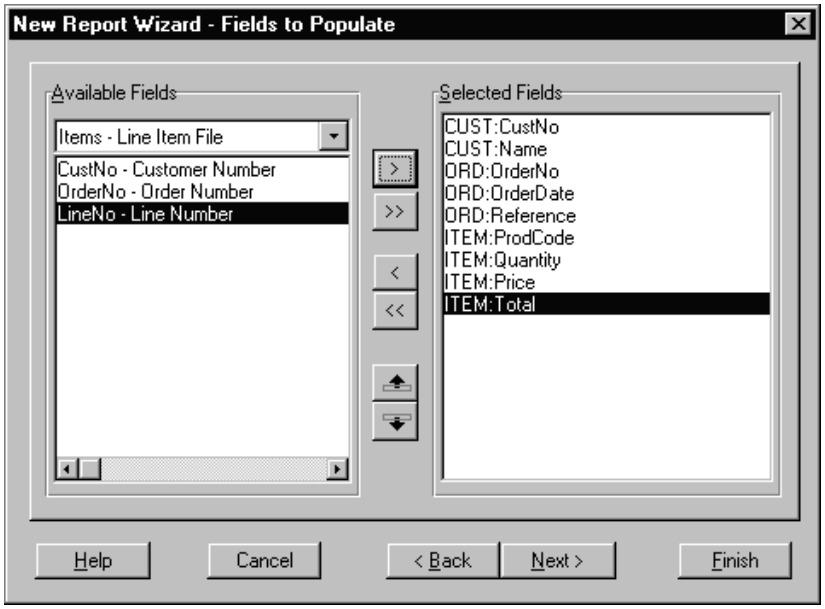
   The first wizard screen appears. This is where you supply the report’s label, a description, select the Report Wizard style and the files the report will use.

2. Add files to the File Schematic by pressing the **Add File…**button.



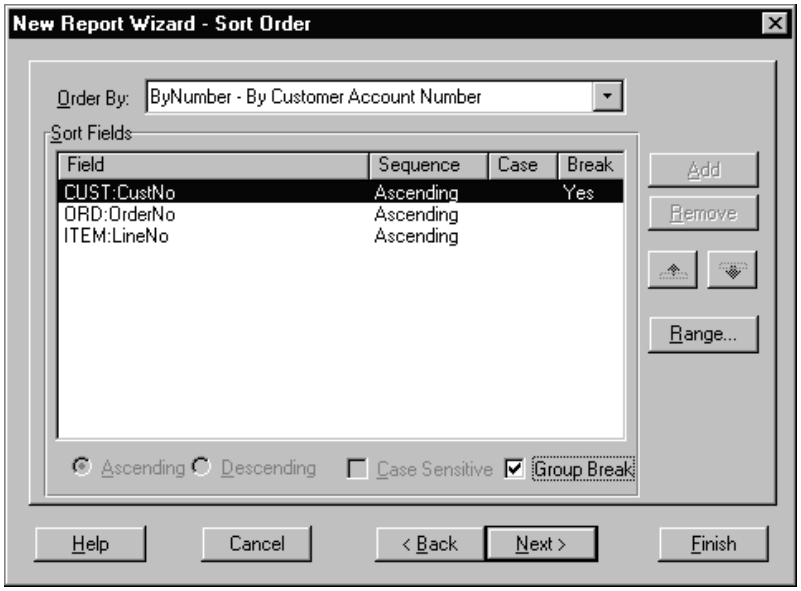
3. Fill in the first wizard window, then press the **Next >** button.

The second wizard window appears. This is where you specify which fields to populate on the report. If you chose the Freeform Wizard, skip to the next section. The Freeform Wizard does not populate any fields.



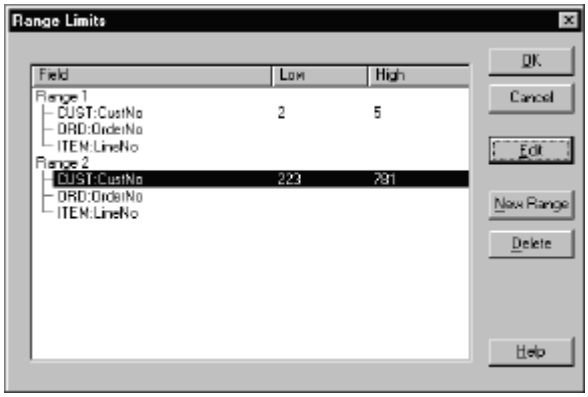
4. Specify the fields to populate, then press the **Next >** button.

   The third wizard window appears. This is where you specify the Sort Order, Range Limits, and Group Breaks for the report.



5. Specify the **Sort Order** and **Group Breaks** for the report.

6. If you want to limit the report's scope, press the Range button and, set the Range Limits you want, then press the OK button.



7. Press the **Next >** button.

The remaining wizard windows depend on which style you chose on the first wizard screen.

Freeform, Table, or Form Wizard

The next wizard window appears. This is where you specify report design options.

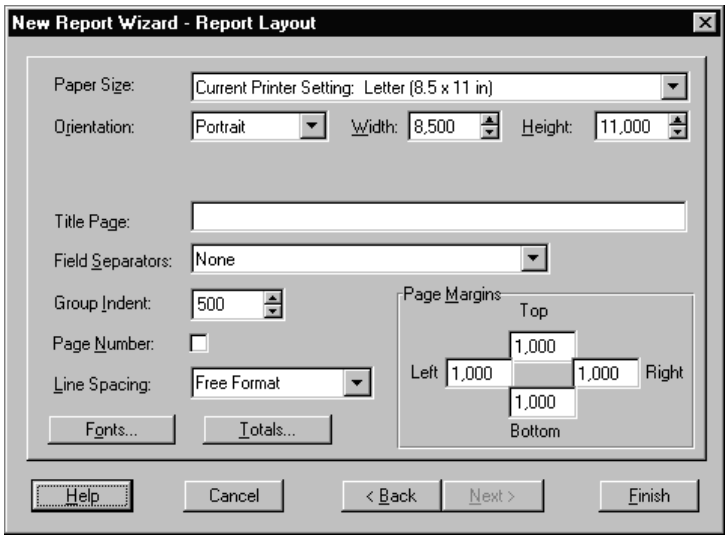
1. Select the **Paper Size** from the drop-down list.
2. Select the **Orientation** from the drop-down list.

   This determines whether the report is Portrait or Landscape.
3. Optionally, set the paper **Width** and **Height**.

   If the paper you are using was in the drop-down list, this is automatically set for you.
4. Optionally, provide the text to place on a title page (if omitted, no title page is created for you).
5. Select the desired **Field Separators** from the drop-down list.

   This determines the type of lines placed on your report.
6. Set the amount of indentation you want for groups using the spin control.

   This determines the amount each level is indented when you create a relational report with group breaks.



7. If you want page numbering, check the **Page Numbering** box.

   This creates a Page Footer and places a page number control in the band.

8. Select the **Line Spacing** from the drop-down list.

   The choices are:

   Free Format
   Single
   1.5 Lines
   Double

   This determines how the controls are populated.

9. Optionally, select a **Font** for elements of the report by pressing the Font button.

   This determines the Font used for controls on each band of the report, unless a font is specified a specific control. Optionally, you can set any of these fonts as the wizard's default.

10. Optionally, select fields to total by pressing the **Totals** button.

    This creates a Grand Total Band and places Total Fields on it.

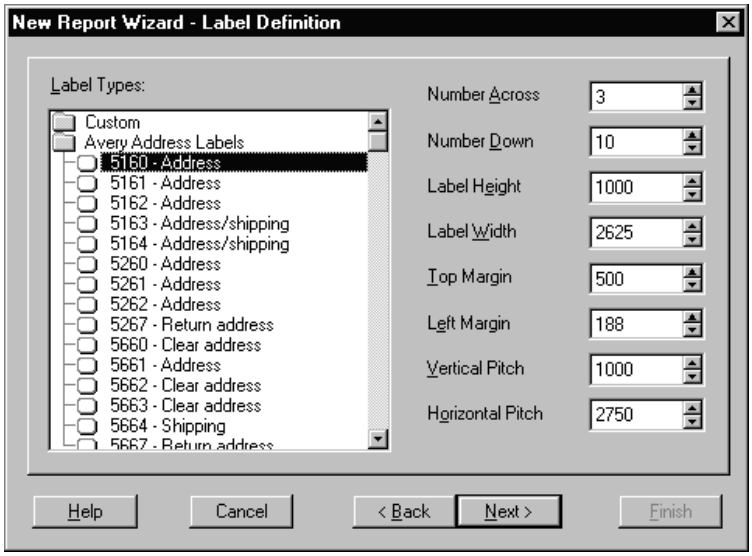
11. Optionally, set the page margins.

12. When you have provided all the information, press the **Finish** button.

    The report appears in the Report Formatter, ready to run or modify.

Label Wizard

The next wizard window appears. This is where you select the type of label.



1. Select the label type from the list.

   This determines the basic size, and configuration of a sheet of labels. If the labels you are using do not appear, design your own by choosing Custom, then define the label specifications on the right.

2. Press the **Next >** button.

   The Report Layout window appears. This allows you to specify the Page Size, Orientation, and Margins.

3. If you want field labels to appear, check the **Prompts** box. If not, clear the check box.

4. Optionally, specify the font to use by pressing the **Fonts** button.

5. Press the **Finish** button.

The report appears in the Report Formatter, ready to run or modify.

Using the Report Formatter

Report Formatter Tools

This section covers the tools available in the Report Formatter. Later sections will explain how to accomplish specific tasks, step-by-step

The Report Formatter Toolbar

The Report Formatter's Toolbar provides easy access to the tools needed most frequently during report design.

Control	What it Does
Page	The Page spin control allows you to jump to any page of a multi-page report.
Band	The Band drop-down list allows you to jump to any band of a multi-band report.
Scale	The Scale drop-down list allows you to resize the formatter's view on your display.
	Calls the File Schematic.
	Calls the Sort Order window.
	Toggles display of the Control Toolbox.
	Toggles display of the Alignment Toolbox.
	Toggles display of the Property List.
	Toggles display of the Field Selection List.
	Runs the report and displays it in Preview Mode.

Report Formatter Toolboxes

ReportWriter provides several toolboxes which you can use. You can toggle a toolbox's display on or off by clicking on the Report Formatter's toolbar, or by choosing the appropriate choice on the View menu. You can also double-click on any toolbox's caption bar to close it.

The Control Toolbox

The Control Toolbox allows you to quickly place controls on a report. click on a control button, then click on the report at the location where the control is to be placed.

Button What it Does

The Cancel Selection Tool returns your cursor to its default state, and cancels a selection. If you click on a Toolbox button and don't want to place a control, clicking on this tool deactivates the selection.

The String Control Tool places a control containing a single line of text. This is equivalent to choosing **Control** ▸ **String Control** from the menu.

The Field Tool places a data field from a file, or a report specific field on a report. This is equivalent to choosing **Control** ▸ **Data Field** from the menu.

The Field Total Wizard Tool quickly creates and places a total field on a report This is equivalent to choosing **Control** ▸ **Field Total** from the menu.

The Text Tool places a multi-line text control from a file, or static text, on a report. This is equivalent to choosing **Control** ▸ **Text Control** from the menu.

The Check Box Tool places a check box on a report, most commonly used to represent a logical field—one that contains a true or false value. If true, the check box prints with an X marking it; if false, the check box prints as a box. This is equivalent to choosing **Control** ▸ **Check Box Field** from the menu.

The Horizontal Line Control Tool places a Horizontal Line graphic control on a report. This is equivalent to choosing **Control** ▸ **Graphic Control** ▸ **Horizontal Line** from the menu.

The Vertical Line Control Tool places a Vertical Line graphic control on a report. This is equivalent to choosing **Control** ▸ **Graphic Control** ▸ **Vertical Line** from the menu.

The Rectangle Control Tool places a rectangular shaped graphic control on a report. This is equivalent to choosing **Control** ▸ **Graphic Control** ▸ **Rectangle** from the menu.

The Ellipse Tool places an elliptically shaped graphic control on a report. This is equivalent to choosing **Control** ▸ **Graphic Control** ▸ **Ellipse** from the menu.

The Group Box Control tool places a grouping box around controls on a report. This is equivalent to choosing **Control** ▸ **Group Box** from the menu.

The Image Control Tool places an image graphic control on a report. This is equivalent to choosing **Control** ▸ **Graphic Control** ▸ **Image** from the menu. This control can display a graphic image from any of the supported graphic file types (.BMP, .JPG, .ICO, .CUR, .WMF, or .GIF).

The Alignment Toolbox

The Alignment Tools allow you to manipulate the spacing and sizing of controls within the report. You can place two or more controls so that their "edges" match up with each other. You can also spread the controls out, or make all of them the same size.

To do so, first select two or more controls. Select the first by clicking on it. Select the second and subsequent controls by pressing the ctrl key, then clicking on the second control while the ctrl key remains pressed.

Button	What it Does
Align Left	Aligns the left borders of the selected controls with the left border of the last control selected (red handles).
Align Right	Aligns the right borders of the selected controls with the right border of the last control selected (red handles).
Align Top	Aligns the top borders of the selected controls with the top border of the last control selected (red handles).
Align Bottom	Aligns the bottom borders of the selected controls with the bottom border of the last control selected (red handles).

Align Horizontally

Along a horizontal axis, aligns the centers of the selected controls with the center of the last control selected (red handles).

Align Vertically

Along a vertical axis, aligns the centers of the selected controls with the center of the last control selected (red handles).

Spread Horizontally

Equalizes the horizontal space between the selected controls.

Spread Vertically

Equalizes the vertical space between the selected controls.

Make Same Size

Makes all selected controls the same height and width as the last control selected (red handles).

Make Same Height

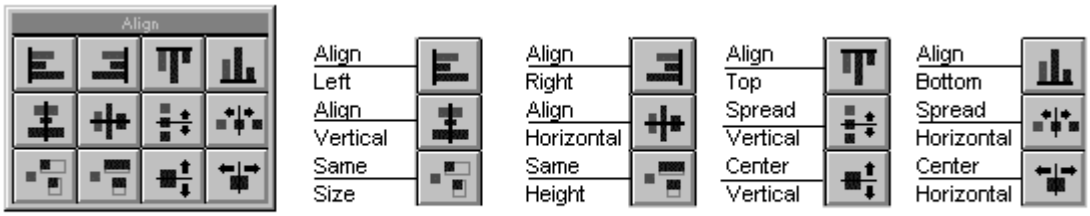
Makes all selected controls the same height as the last control selected (red handles).

Center Horizontally

Centers the selected controls horizontally in the band.

Center Vertically

Centers the selected controls vertically in the band.



Field Selection List

The Field Selection List allows you to populate Data Field controls quickly. This Tool provides access to all the files in the Report's File Schematic.

To populate a control using the Field Selection List:

1. Select the file from the drop-down list.
2. DOUBLE-CLICK on the desired field.
3. CLICK on the report at the location where you want to place the field.

The Property List

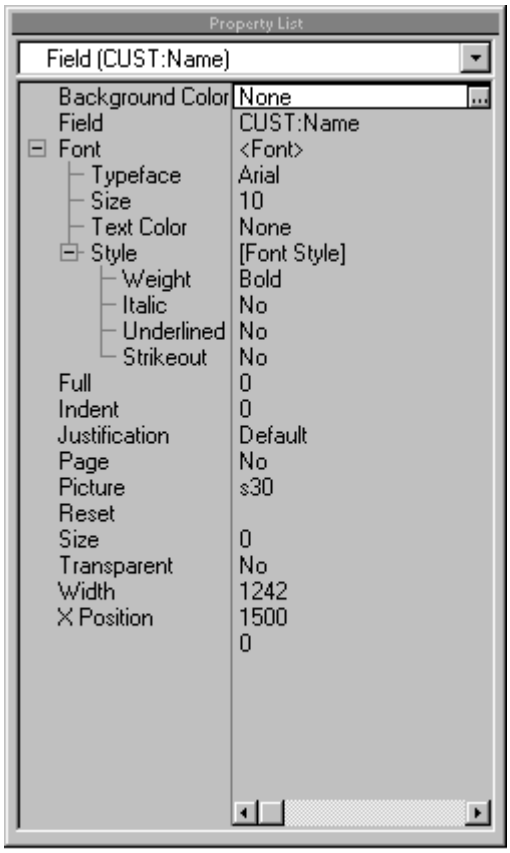
Bands and Control behavior is defined through their properties. Each Band type and Control type has its own set of properties. By specifying properties you can, for example, print a report's title in a specified font or ensure that a group header band prints on the same page as its group details.

Properties are specified through the Property List. This list changes depending on the selected item (control or band). Control and Band Properties are explained in detail later in this chapter.

There are two ways to access the Property List. You can enable the "floating" Property list by pressing the  button on the toolbar or you can access the Property List for a single element by right-clicking on it and selecting **Properties**. When using the floating Property List, you do not need to close the list after each modification. It remains available to modify the currently selected element of your report.

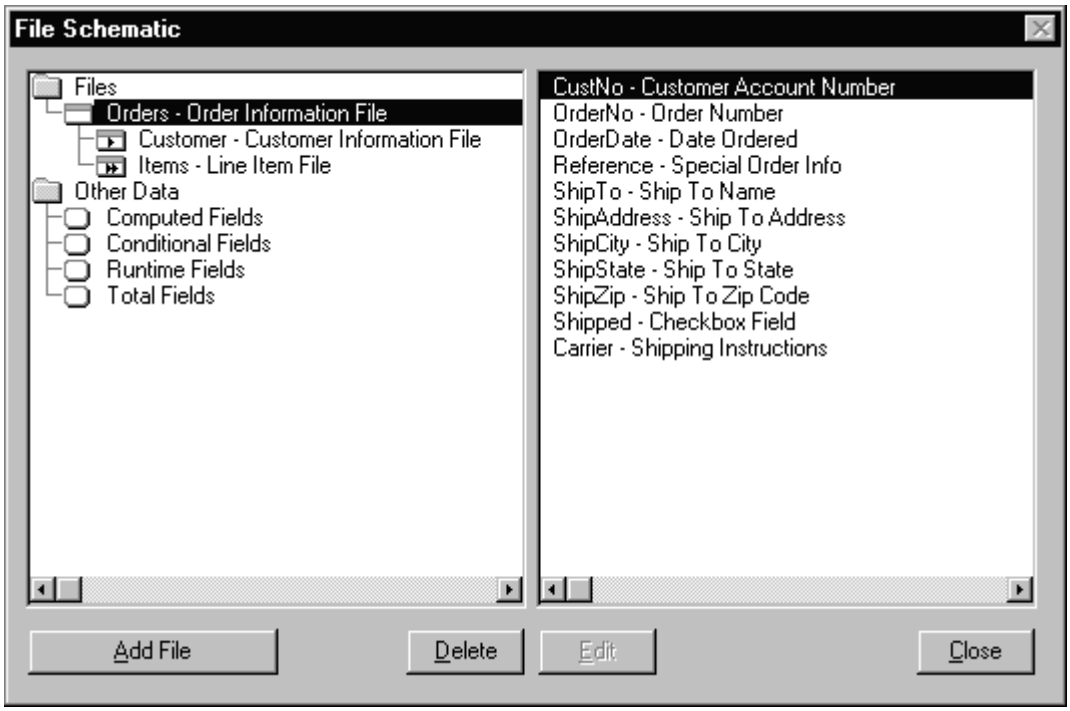
To set an item's properties:

1. Select the item by clicking on it or selecting it from the Property List's drop-down list.
2. Modify the properties as needed.



The File Schematic

Each report is based on a collection of data files. The report's file schematic is created when you specify files in the Report Wizard. You can modify the File Schematic after the report is created by pressing the  button on the toolbar or choosing **Report** ▸ **File Schematic**.



Non-Printing Guides— Rulers, Grid, and Guides

Non-printing guides help you position items on a report, but do not print on the report. These lines form the framework of the layout grid. There are three kinds of non-printing guides: Rulers, the layout grid, and Horizontal and Vertical Guides.

Rulers

There are ruled lines at the top and bottom of the formatter. The rulers measure in inches or centimeters, depending on the unit of measurement on which the report is based. Rulers can be useful in lining up items on a report so columns can be kept straight. As you move the mouse, tick marks appear on the rulers which allow you to see the exact position.

Horizontal and Vertical Guides

In addition to the rulers, you can use guides to assist in aligning items on a report. These guides are created by clicking on a ruler. If you click on the horizontal ruler at the top of the formatter, you'll get a Vertical guide at that position. If you click on the vertical on the left side of the formatter, you'll get a Horizontal guide at that position. After a guide is created, you can drag it to any desired location. To remove a guide, merely drag it to the ruler and discard it.

Grid

When you position controls in the report, you can then optionally force them to align on grid points using the Report Grid.

To toggle the Grid on or off:

1. Choose **View** ▸ **Grid**.

Files, Sort Order, and Range Limits

Specifying the File Schematic for a Report

The File Schematic specifies the files used for the report. Reports use one Primary file and multiple levels of related files.

The File Schematic also lists the available report-specific fields. Report-specific fields are the computed, conditional, total, or runtime fields that have been defined for a report. Report-specific means they are only available for the report in which they are defined. See Chapter 9 for information on creating report-specific fields.

Before specifying a file schematic, decide which file will be the primary file and which related files you will need for the report.

1. CLICK on the  button or Choose **Report** ▸ **File Schematic**.

   The File Schematic Definition window appears. This contains a list box which displays all the files used in a report. The list box displays in a tree format with two sections: *Files* and *Other Data*.

2. With the *File Schematic* band (in the **Files** list box) highlighted, press the **Add File** button.

   The Select Primary File window appears, displaying a list of all the files in the Report Library.

3. Highlight the file and press **Select**.

   If you are using only a primary file, you can skip the remaining steps.

4. Highlight the file to which you want to attach a related file.

5. Press the **Add File** button.

   The Select File window appears. This window displays the files that are related to the Primary file (defined in the Data Dictionary on which the Report Library is based) in the Related Files section.

6. Highlight the related file in the Related Files section and press the **Select** button.

   The file schematic now displays the files and their relationships. The top file is the Primary file, any other files are Secondary files.

   Secondary files are directly related to the file above it in the schematic. A double right arrow indicates that the Secondary file is the Child of the file to which it is connected. A single right arrow indicates that the Secondary File is the Parent of the file to which it is connected.

   If you want to add more related files, repeat the last step.

7. When all the files are in the schematic, press the **OK** button.

Creating a User Defined Relationship

Files can be joined in a Data Dictionary if the files contain the necessary keys. If these relationships have not been defined in a Data Dictionary or your report is not based on a Data Dictionary, you can create a user-defined relationship in a report's file schematic.

This is particularly useful when joining files created with file systems that do not support keys (such as ASCII or BASIC).

The two files must have fields which contain identical data to use as a link between them. The files must be in the File Schematic specified for the report.

Before specifying a file schematic, choose the primary file and identify the related files needed for the report.

1. CLICK on the  button or Choose **Report** ▸ **File Schematic**.

   The *File Schematic Definition* window appears. This contains a list box which displays all the files used in a report. The list box is divided in two sections: *Files* and *Other Data*.

2. With the File Schematic band highlighted, press the *Add File* button.

   The *Select Primary File* window appears, displaying a list of all the files in the selected dictionary.

3. Highlight the desired file and press **Select**.

4. Highlight the file from which to attach another file.

5. Press the **Add File** button.

   The *Select File* window appears. This window has 2 sections: Related Files and User-defined Related Files. The first section displays the files that have defined relations to the Primary file in the Database Dictionary. The second lists all the remaining files in the Report Library.

6. Highlight the related file in the *User Defined Related Files* section and press the **Select** button.

   The *User Defined Relationship Properties* window appears.

7. Select either the **1 to Many** or **Many to 1** button.

8. Press the **Add** button.

   The *Link Fields* window appears. This window contains two drop-down lists containing all the fields in the two files.

9. Select the Link From field from the drop-down list.

10. Select the Link To field from the drop-down list.

11. Check the **Case Sensitive** box, if you don't want to ignore case.

Note:

When a field contains numeric data, you always want to consider it in a Case Sensitive manner. When case is ignored, numbers are sorted alphabetically.

12. Press the **OK** button.

    The link appears in the Link Fields list box in the User Defined Relationship Properties window.

13. For additional links, repeat the last 5 steps.

14. Press the **OK** button.

    The file schematic now displays the files and the user-defined relationship. The top file is the Primary file, any other files are Secondary files.

    Secondary files are directly related to the file above it in the schematic. A double right arrow indicates that the Secondary file is the Child of the file to which it is connected. A single right arrow indicates that the Secondary File is the Parent of the file to which it is connected.

15. When all the files are in the schematic, press the **Close** button.

Specifying the Report Order

1. CLICK on the  button or Choose **Report** ▸ **Sort Order**.

   The Sort Order window appears.

2. Select the desired Sort Order type from the drop-down list in the Order By field.

   The drop-down list contains the keys defined in the Data Dictionary, Record Order, and one for USER DEFINED ORDER. If no defined key fits the sort order, the USER DEFINED ORDER allows you to create a sort order used only by this report (see *Creating a User Defined Order*).

Note:

If you change the Sort Key for a report containing Range Limits, you will lose any Range Limits you have set.

3. Optionally, specify any group breaks by highlighting the field to break on and checking the **Group Break** Box beneath the list.

4. Press the **OK** button.

Creating a User Defined Order

If no key in the Database Dictionary fits the sort criteria you would like to use, you can create your own. This allows you to sort the data on any field.

1. CLICK on the  button or Choose **Report** ▸ **Sort Order**.

   The *Sort Order* window appears.

2. Select the *User Defined Order* from the drop-down list in the **Order By** field.

   The drop-down list contains the keys defined in the Data Dictionary, Record Order, and one for USER DEFINED ORDER. If you find that an existing key fits the sort criteria, you can select that one. If not, you can continue creating your own.

3. Press the **Add** button.

   The *Fields Selection* window appears.

4. Select the sort field from the **Fields** list box, then press the **Select** button. You can select from any file in the file schematic.

5. With the field highlighted, check the appropriate boxes to specify the properties of the user-defined sort. The choices are:

   Ascending Mark this button to specify the records are to be sorted from lowest to highest.

   Descending Mark this button to specify the records are to be sorted from highest to lowest.

   Case Sensitive If checked, ReportWriter performs a case sensitive sort. If not checked, case is ignored.

   Note:

   When a field contains numeric data, you always want to consider it in a Case Sensitive manner. When case is ignored, numbers are sorted alphabetically.

   Group Break Check this box to cause a group break on this field. (See *Defining Group Breaks*)

Defining Group Breaks

Group breaks determine how data is grouped together in a report. Grouping data in different ways produces reports that can provide valuable business information.

For example, if you set a group break on the STATE field, data would print out with all records with the same state name grouped together. If you add a group break on the City field, you could have all the records with the same city name grouped together within the state grouping.

The Group Headers and Group Footer bands allow a place for titles, subtotals, and totals. Using these group breaks, you could print a sales report by city and state, with totals for each city and totals for each state.

Once you have determined the grouping you want for your report, you can set the group breaks.

1. CLICK on the  button or Choose **Report** ▸ **Sort Order**.

   The *Report Sort* window appears, displaying a list of all the fields that are available for group breaks. These fields are all components of the keys which link the files. Only linking fields can be used as group break fields.

2. Highlight the field to break on and check the **Group Break** box.

   If you want more than one group break on the report, repeat the last step for each field the report should break on.

3. Press the **OK** button.

   The Report Formatter now shows bands for the Group Header for each break field specified. You can also add Group Footer bands for any of these groups breaks.

Using Range Limits

You can reduce processing time by using a subset of the data file that only contains the records you need. Range Limits define this subset. ReportWriter allows you to define multiple ranges, so that you can access data within more than one range.

A simple range limit could, for example, limit the scope to access only those records from one state. Using a multiple range, you could access the records for two states. You can also select a Runtime Field as a Range Limit Value.

1. CLICK on the  button or Choose **Report** ▸ **Sort Order**.

   The *Report Order* window appears.

2. Press the **Ranges...** button.

3. Highlight the field to limit, then press the **Edit** button.

   The *Range Limits* window appears.

4. Type the low limit value for the range to consider in the **Enter Low Value** field.

Selecting the **No Low Value** radio button specifies that the low-end of the limit is the lowest possible value for the field's data type.

5. Type the low limit value for the range to consider in the **Enter High Value** field.

   Selecting the **No High Value** radio button specifies that the high-end of the limit is the highest possible value for the field's data type.

6. Press the **OK** button.

7. For additional ranges, press the **Insert Range** button and repeat the last three steps.

Filtering a Report

Records can be filtered so they only print if they meet a certain criteria. This produces a report containing only the information you want.

Records can be filtered in many different ways. For example, you might want a report of "widget" sales in 1989, but only if the total sale exceeded $1,000. The file contains sales data for many products and several years of sales.

Using the filter below, only those records print.

Year = 1989 AND Item = 'Widget' AND Sale > 1000.00

To set a record filter:

1. CLICK on the  button or Choose **Report** ▸ **Record Filter**.

   The *Record Filter* window appears with a built-in Formula Editor. This allows you to create a syntactically correct expression. With the Formula Editor, you add components one-by-one from a list of the available components. The Formula Editor offers only valid options every step of the way.

   Since a record filter requires that a certain condition be met, the Formula Editor creates a Conditional expression. A conditional expression is evaluated as true or false. For example, the expression: Date < 06/07/1996 is true if the value of the Date field is 06/06/1996 or less. If it is greater than or equal to 06/07/1996, then the expression is false.

2. Create a condition with the *Formula Editor* (see *Using the Formula Editor*).

3. Press the **OK** button.

   ReportWriter now checks the condition as each record is read and only processes those that evaluate true.

Conditional Bands

You can place a condition on a band so it prints only if a certain criteria is met.

For example, you might want to print one band for customers in your state, and a different band for out-of-state customers. You would create a second detail band, set a band condition on each— one to print if State = YourState, the other to print if State <> YourState.

To set a Band Print filter:

1. Select the band.
2. Open the Property List (by clicking on the  button on the toolbar).
3. CLICK on the Condition property, then press the ellipsis (...) button next to *Condition*.

   The Formula Editor appears. The Formula Editor enables you to create a syntactically correct expression. In the Formula Editor, you add components one-by-one from a list of the available components. The Formula Editor offers only valid options every step of the way.

   Since a band print filter checks if a certain condition is met, the Formula Editor creates a Conditional expression. A conditional expression evaluates as true or false. For example, the expression: Date < 06/07/1996 is true if the value of the Date field is 06/06/1996 or less. If it is greater than or equal to 06/07/1996, then the expression is false.
4. Create a condition with the Formula Editor (see *Using the Formula Editor*).
5. Press the **OK** buttons on the *Generate Expression* and the *Band Properties* windows.

   ReportWriter now evaluates the condition before printing the band and only prints it when the condition is true.

Placing Fields on a Report

Before placing a field on a report, you must first decide which band it should be placed in. Once you have decided, you are ready to continue.

1. Choose **Control** ▸ **Data Field**.

   The File Schematic for the report displays. When creating a new report, the file(s) used must be placed in the file schematic (see *Specifying the File Schematic*).
2. If the desired file is already highlighted in the file schematic, skip this step. If not, highlight the desired file in the Files list box.

   The file's available fields appear in the Fields list box.
3. Highlight the field you want to place on the report and press the **Select** button (or DOUBLE-CLICK on the field).
4. CLICK on the report at the location where you want to place the field.

Placing Image Fields on a Report

ReportWriter supports printing graphic images from several sources.

The image's source can be a file or a binary memo or BLOB field which contains the image data. If a Memo or BLOB contains the data for a graphic, ReportWriter automatically interprets its format and prints the image. This allows you to store any supported graphic in a binary memo or BLOB and print the image it contains.

The following graphic formats are supported:

Graphical Interchange Format	**.GIF**
	.JPG
Bitmap Graphics	**.BMP**
Windows MetaFile	**.WMF**
PaintBrush Format	**.PCX**
Icon File	**.ICO or .CUR**

To Place an Image on a Report:

1. Choose **Control** ▸ **Graphic Control** ▸ **Image**.
2. CLICK on the report at the location where you want the image to appear.
3. Specify the Image Source.
4. If you want the image to scale to the correct proportions, check the **Maintain Aspect Ratio** box.

   This sets the Height Property to the size of the control and the Width property to 0. These settings mean the IMAGE will resize the width to scale the image properly to the Height depending on the image source.
5. Press the **OK** button.

Moving Controls on a Report

The Report Formatter allows you to move items easily. All you have to do is click-and-drag with the mouse. In addition, if a control is selected, you can "nudge" it with the cursor keys. Each press of a cursor arrow key moves the control in the direction of the arrow.

Copying, Cutting, and Pasting Controls on a Report

The Report Formatter allows you to copy or cut items and paste them into any band. This allows you to move a control from one band to another. If you want to move a control to a new band, cut it, them paste it in the desired band.

Deleting Controls from a Report

1. Select the control.
2. Press DELETE.

Auto-Populating a Band on a Report

If you want to place several fields from a data file on a report quickly, the Auto-Populate method is the answer. Using this method, data fields are placed side-by-side. The fields can be moved afterwards, if you choose.

1. Choose **Edit** ▸ **Auto-Populate**.

   The *Field Selection* window appears.

2. If the file is already selected in the drop-down list, skip this step. If not, select the file in the File drop-down list.

   The file's available fields appear in the Fields list box.

3. Highlight the field you want to place on the report and press the  button (or double-click on the field). If you want to populate all the fields, press the  button.

   The Data Field properties are set to the defaults from the Database Dictionary. If you want to change them, you can edit them later.

4. Specify the direction you want the fields to populate—**Horizontal** places controls side-by-side; **Vertical** places them in a column.
5. Select the **Band to Populate** from the drop-down list.
6. Press the **OK** button.

   All the fields are placed on the report. You can move any field(s) or edit their properties, if you wish.

Report Layout

Controlling Pagination

There are four properties which control the manner in which report elements span pages (pagination). Using these pagination options, you can create a report that prints in an organized manner and ensures that every page is understandable.

Page Before Print the structure on a new page. To reset the page number to a value you specify, type it in the Page Before field. To force a Page Before without resetting the page number, specify -1.

Page After Print the structure, then force a new page. To reset the page number to a value you specify, type it in the Page After field. To force a Page After without resetting the page number, specify -1.

With Prior Prevents 'orphan' elements in a printout. An 'orphaned' print element is one which prints on a following page, separated from its related items. The value specifies the number of preceding elements to print—a value of "1," for example, specifies that the previous element must print on the same page. When placing subtotals or totals in a band, use the WITHPRIOR attribute to insure they print with at least one member of the column above it when a page break occurs.

With Next Prevent 'widow' elements in a printout. A 'widowed' print element is one which prints, but then is separated from the succeeding elements by a page break. The value specifies the number of succeeding elements to print—a value of '1,' for examples, specifies that the next element must print on the same page, else page overflow puts them both on the next.

You can use almost any combination of these properties to control the report's pagination.

Variable Length Memos on Reports

Memo fields are usually used to store data of varying lengths. One record may contain several lines of text in its Memo, while another may contain a few words. Typically, you will want your report to adjust to the size of the Memo. If it were a fixed size, you would have blank areas on the report.

The Report Wizard automatically creates bands to handle the varying length of a Memo Field. The report should not contain noticeable "gaps."

This is accomplished with a combination of two property settings. First the TEXT control for the Memo field is set to **Auto Expand**. This specifies that the Text control expands to accommodate all the data in the memo. The second property is on the Group Header Band. It is set to not be a **Fixed Height**. This allows the band to expand and contract to accommodate the size of the controls within the band.

To set the properties used to produce this effect:

1. RIGHT-CLICK on the text control for the Memo, then select **Properties**.
2. Set **Auto Expand** to *Yes*.
3. Press the **Close** button.
4. RIGHT-CLICK on Band, then select **Properties**.
5. Set **Fixed Height** to *No*.
6. Press the **Close** button.

Using these two properties, you can create expanding Memos and Bands in any report.

Placing Text on a Report

Constant or literal text can be used as report titles, page headers or footers, column headings, group headers, or report final page footers. Text can be placed on a report by using either the String or the Text Control.

Lines, Boxes, Circles

Lines, boxes, and ellipses can be used to separate groups or place borders around certain items to make them stand out. Good use of these controls improves the report's readability.

To Place a Line on a Report:

1. Choose **Control** ▸ **Graphic Control** ▸ **Horizontal Line or Control** ▸ **Graphic Control** ▸ **Vertical Line**
2. CLICK on the report at the location where you want the line to appear.
3. Adjust the line length by dragging the handles, as needed.

The Picture Editor

Some controls let you specify a picture token that provides a special format for displaying and printing variables.

There is a great variety and diversity of picture token syntax which depends on the type of data you format: strings, numbers, currency, scientific, dates, times, etc.

The *Edit Picture* dialog lets you quickly and easily build an appropriate picture token without memorizing picture token syntax. Invoke this easy to use dialog by pressing the ellipsis (...) button beside the **Picture** prompt.

Example An example of the print or display format currently specified in the dialog. What you see is what you get.

Picture The picture token currently specified. This picture token produces the example above.

Picture Type Choose the type of data to format from this drop-down list. Choose from String, Numeric and Currency, Scientific Notation, Date, Time, Pattern, and Key-in Template.

String

String picture tokens specify a length, with no other formatting.

Length Specify the length of the string. This length also determines the width of the control if the width is not otherwise specified.

Numeric and Currency

Numeric and currency picture tokens specify a length, plus conventional formatting to convey positive and negative values, various currencies, etc. For more information see the *Print and Display Formatting* appendix.

Size The total number of digits, plus any formatting characters. For example, $22.25- is 4 digits + 3 formatting characters for a size of 7.

Decimal Digits The number of digits to the right of the decimal.

Currency Choose from None, Leading, and Trailing. None shows no currency symbol. Leading puts the currency symbol to the left of the number and Trailing puts the symbol right of the number.

Symbol The currency symbol to print or display: either a dollar sign ($) or a string constant.

Negative Sign Specify how negative values print.

- **Bracket** Negatives surrounded by parentheses.
- **Leading** Negatives get a leading minus sign.
- **Trailing** Negatives get a trailing minus sign.
- **None** No negative sign prints or displays.

Decimal Separator

Specify the character inserted between the integer and fractional portion of the value.

Period Period is the separator.

Comma Comma is the separator.

None No separator prints or displays.

Grouping Specify the character inserted at every third digit to aid readability.

Comma Comma is the separator.

Period Period is the separator.

Space Space is the separator.

Hyphen Hyphen is the separator.

Leading Character

Specify the character to represent leading zeroes.

Clip Remove leading zeroes so that any leading format characters abut the left most digit.

Zero Leading zeroes print or display as zeroes (0).

Space Leading zeroes print or display as spaces ( ).

Asterisk Leading zeroes print or display as asterisks (*).

Blank when zero

Check this box to print or display nothing when the value is zero.

Scientific

Scientific Notation picture tokens let you print or display very large or very small numbers with a decimal format raised by a power of ten. The print or display format takes the form -9.99e+999. For more information see the Print and Display Formatting appendix.

Number of Characters

The total number of characters, including the 7 format characters. For example, -1.96e+007 requires 10 characters.

Leading Digits The number of digits to the left of the decimal point (typically 1).

Decimal Separator

Specify the character inserted at every third digit to aid readability.

Period Period is the separator.

Comma Comma is the separator.

Space A Space is the separator.

Separator Specify the character inserted between the integer and fractional portion of the value.

Period Period is the separator.

Comma Comma is the separator.

Blank when zero
Check this box to print or display nothing when the value is zero.

Date

Date Notation picture tokens let you print or display dates in a number of different formats. Choose the format you want from the Format drop-down list. For more information see the *Print and Display Formatting* appendix.

Format

Choose the format you want from the drop-down list. What you see is what you get except for the Windows Short and Windows Long formats. Additionally, the separator character and leading zeroes may be specified independent of the chosen format.

Windows Short

Uses the short date format specified in Windows Control Panel or Windows 95 Regional Settings control panel.

Windows Long

Uses the long date format specified in Windows Control Panel or Windows 95 Regional Settings control panel.

Separator Choose from Standard (/), Period (.), Dash (-), Space ( ), and Comma (,).

Leading Character Specify the character to represent leading zeroes.

Zero Leading zeroes print or display as zeroes (0).

Blank Remove leading zeroes.

Asterisk Leading zeroes print or display as asterisks (*).

Blank when zero Check this box to print or display nothing when the value is zero.

Time

Time Notation picture tokens let you print or display times in a number of different formats. Choose the format you want from the Format drop-down list. For more information see the Print and Display Formatting appendix.

Format

Choose the format you want from the drop-down list. What you see is what you get except for the Windows Short and Windows Long formats. Additionally, the separator character and leading zeroes may be specified independent of the chosen format.

Windows Short

Uses the time format specified in Windows Control Panel or Windows 95 Regional Settings control panel.

Windows Long

Uses the time format specified in Windows Control Panel or Windows 95 Regional Settings control panel.

Separator	Choose from Standard (:), Period (.), Dash (-), Space ( ), and Comma (,).
Leading Character	Specify the character to represent leading zeroes.
Zero	Leading zeroes print or display as zeroes (0).
Blank	Remove leading zeroes.
Blank when zero	Check this box to print or display nothing when the value is zero.

Pattern

Pattern picture tokens let you build custom print or display formats for various numbers: phone numbers, social security numbers, room numbers, dates, times, measurements, etc. For more information see the Print and Display Formatting appendix.

Picture	Type the picture token between the 'P's according to the Legend below. Your picture token can include any displayable characters, including all the standard keyboard characters. At runtime, the constants in the picture token display just as they appear in the token. The left angle (<) and the pound sign (#) resolve into the individual digits from the display variable.
Legend	< integer, blank if zero # integer constant (any displayable char except < and #).
Blank when zero	Check this box to print or display nothing when the value is zero.

If you want to use a lowercase p in your picture, use an uppercase P at the start and end of your picture token. If you want to use an uppercase P in your picture, use a lowercase p at the start and end of your picture token.

Key-in Template

Pattern picture tokens let you build custom edit formats for STRINGs, CSTRINGs and PSTRINGs containing mixed alphanumeric characters. Although Key-in tokens affect output as well as input, their primary purpose is to provide custom field editing and validation on input when using Runtime fields.

Picture Type the picture token between the 'K's according to the Legend below. Your picture token can include any characters (displayable or not), including all the standard keyboard characters.

Legend

< accept an integer, blank if zero
accept an integer
? accept any character (even non-display)
^ accept an upper case character
_ accept a lower case character
| input may stop here
constant (any displayable char except <#?^_| or \)
\ display next char (lets you display <#?^_| and \)

Only alphabetic characters

Check this box to accept only alphabetic characters.

Blank when zero

Check this box to print or display nothing when the value is zero.

If you want to use a lowercase k in your picture, use an uppercase K at the start and end of your picture token. If you want to use an uppercase K in your picture, use a lowercase k at the start and end of your picture token.

For more information see the *Print and Display Formatting* appendix.

9 - RUNTIME , TOTAL, COMPUTED AND CONDITIONAL FIELDS

Report-Specific Fields

Runtime , Total, Computed, and Conditional fields are report-specific—fields used only by one report. Report-specific fields can be used alone or in combination with other fields in a formula. There are four types of report-specific fields:

Total Field:	Calculates a sum, average, count, minimum, or maximum on a specified data field and assigns that value to the total field.
Computed Field:	Performs a specified calculation on data and assigns the result to the field.
Conditional Field:	Evaluates an expression and assigns one value to the field if the expression is true and another if the expression is false.
Runtime Field:	Accepts input from the user at runtime, then assigns the value to the runtime field.

Total Fields

There are five types of “total” fields: Sum, Average, Count, Max, and Min.

A **Sum** field keeps a running total of the field selected as the Use Field. An **Average** field totals the Use Field and divides that sum by the number of fields processed, resulting in the arithmetic mean (average value) of that field. A **Count** field tallies the number of times the Use Field is printed. A **Min** or **Max** field evaluates and stores the minimum or maximum values of the Use Field, respectively.

There are two methods to create a Total Field—the Total Wizard and the Standard Method.

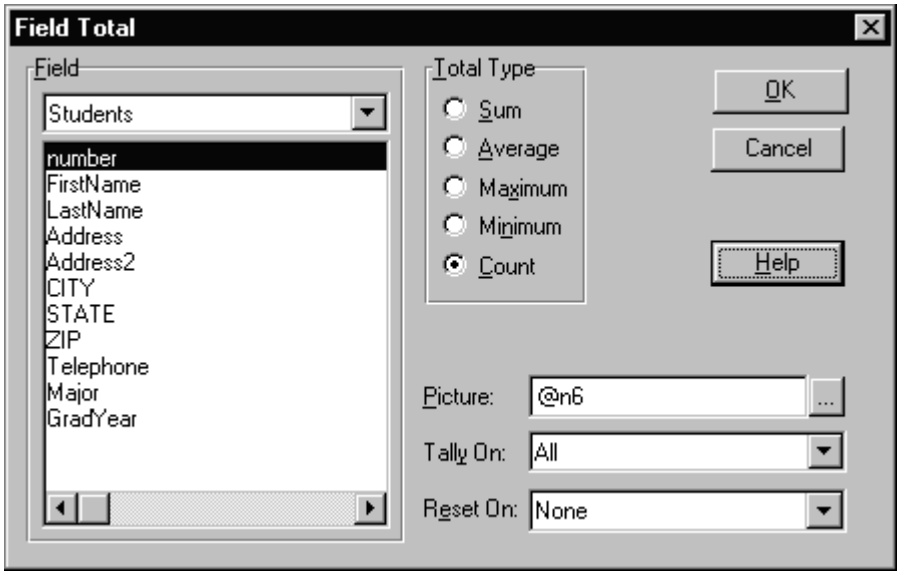
Creating Total Fields (Total Wizard)

1. Choose **Control** ▸ **Field Total** (or press the Σ button on the Control toolbox).

   The *Field Total Wizard* appears. This wizard lets you specify the field on which to total, the total type and how it is tallied and reset.

2. Select the field to total from the list on the left. If necessary, change the file by selecting a different one from the drop-down list.

3. Select the total **Type** from the drop-down list (*Sum*, *Average*, *Count*, *Max*, or *Min*).



4. Type the picture for the field in the **Picture** field, or press the ellipsis (...) button to use the Picture Editor.

   This specifies how the total field's value displays. See the *Print and Display Formatting* appendix for more information on Pictures.

5. Select the **Tally On** choice from the drop-down list.

   The choices are PAGE, ALL, or any Group Breaks defined in the report. If you choose PAGE, the total computes every time page overflow occurs. If you choose ALL, the total computes every time the Detail Band prints. If you choose a Group Break, the total computes every time the group break occurs.

6. Select the **Reset On** choice from the drop-down list.

   The choices are PAGE, NONE, and any Group Breaks defined in the report. If you choose PAGE, the total resets every time page overflow occurs. If you choose NONE, the total resets only when the report is completed. If you choose a Group Break, the total resets every time the value of the group break field changes.

Note:

Your Tally On and Reset On choices should never be the same. If they are the same, you have told ReportWriter to tally and reset the total at the same time—the total will always be zero.

7. Press the **OK** button.

8. CLICK on the Report at the location where you want the total field.

Creating Total Fields (Standard Method)

1. Choose **Report** ▸ **File Schematic** (or press the  button on the toolbar).

   The *File Schematic* window appears. The File Schematic lists all of the report's files, the fields in those files, and the report-specific fields.

2. Highlight *Total Fields* in the File Schematic, then press the **Add Total Field** button.

   The *Total Field* window appears. This is where you define the total field.

3. Type a name for the field in the **Label** field.

   The Label can contain only alphanumeric characters (letters or numbers) or the underscore (_) character, and must begin with a letter. Spaces are not allowed. The maximum length is 40 characters. Each field name must be unique. There can be no duplicate names within a report.

4. Optionally, provide a **Description**.

   A description can help you identify the total field.

5. Type the picture for the field in the **Picture** field, or press the ellipsis (...) button to use the Picture Editor.

   This specifies how the total field's value displays. See the *Print and Display Formatting* appendix for more information on Pictures.

6. Select the **Field to Total** by pressing the ellipsis (...) button.

   The *Field Selection* window appears. Highlight the field on which to base the calculation, then press the **Select** button.

7. Select the **Tally On** choice from the drop-down list box.

   The choices are PAGE, ALL, or any Group Breaks defined in the report. If you choose PAGE, the total computes every time page overflow occurs. If you choose ALL, the total computes every time the Detail Band prints. If you choose a Group Break, the total computes every time the group break occurs.

8. Select the **Reset On** choice from the drop-down list box.

   The choices are PAGE, NONE, and any Group Breaks defined in the report. If you choose PAGE, the total resets every time page overflow occurs. If you choose NONE, the total resets only when the report is completed. If you choose a Group Break, the total resets every time the value of the group break field changes.

   Note:

   Your Tally On and Reset On choices should never be the same. If they are the same, you have told ReportWriter to tally and reset the total at the same time—the total will always be zero.

9. Select the total **type** from the drop-down list (*Sum, Average, Count, Max,* or *Min*).

10. Press the **OK** button.

    The total field appears in the file schematic and you can place it on the report.

11. Press the **Close** button to exit the File Schematic.

To place a Total field on a report:

1. Choose **Control** ▸ **Data Field**.

   The *File Schematic* window appears.

2. Highlight *Total Fields* in the File Schematic.

3. Highlight the Total Field in the Fields List, then press the **Select** button.

4. CLICK on the report at the location to place the field.

Computed Fields

A computed field is a field whose value is determined by the result of an expression. This is similar to an assignment statement in most programming languages. The computed field is assigned the value from the expression.

Expressions can range from the very simple to very complex. For example a computed field to return today's date uses the formula:

    TODAY()

For a computed field to calculate the tax for an invoice, one possible formula might be

    SubTotal * TaxRate.

All of these expressions are created in the Formula Editor, which produces the correct syntax for the expression.

To create a Computed Field:

1. Choose **Report** ▸ **File Schematic** (or press the  button on the toolbar).

   The *File Schematic* window appears. The File Schematic lists all of the report's files, the fields in those files, and the report-specific fields.

2. Highlight *Computed Fields* in the File Schematic, then press the **Add Computed Field** button.

   The Computed Field window appears. This is where you define the computed field.

3. Type a name for the field in the **Label** field.

   The Label can contain only alphanumeric characters (letters or numbers) or the underscore (_) character, and must begin with a letter. Blank spaces are not allowed. The maximum length is 40 characters. Each field name must be unique. There can be no duplicate names within a report.

4. Optionally, provide a **Description**.

   A description can help you identify the Computed field.

5. Type the picture for the field in the **Picture** field, or press the ellipsis (...) button to use the Picture Editor.

   This specifies how the total field's value displays. See the *Print and Display Formatting* appendix for more information on Pictures.

6. Type a Formula in the entry box, or press the **Formula** button to use the *Formula Editor*.

   The Formula Editor appears. This lets you create a syntactically correct expression. In the Formula Editor, you add components one-by-one from a list of the available components. The Formula Editor offers only valid options every step of the way.

7. Create an expression with the Formula Editor (see *Using the Formula Editor*).

8. Press the **OK** button.

To place a Computed Field on a report:

1. Choose **Control** ▸ **Data Field**.

   The *File Schematic* window appears.

2. Highlight *Computed Fields* in the File Schematic.

3. Highlight the Computed Field in the **Fields List**, then press the **Select** button.

4. CLICK on the report at the location where the field is to be placed.

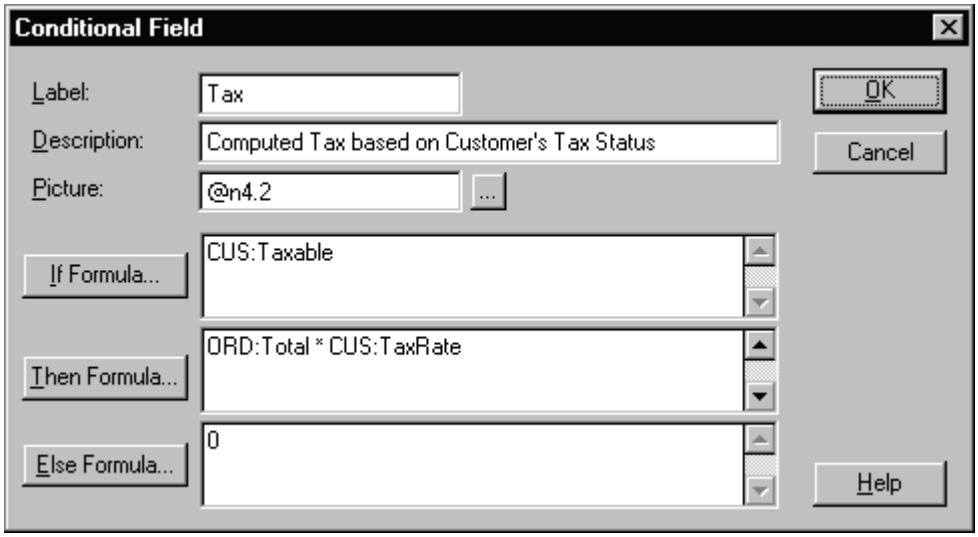
Conditional Fields

A conditional field is assigned one of two possible values: one if the condition is true; the other if the condition is false. To create a conditional field, first determine the condition to evaluate, then determine the values to assign to the field when the condition is true and when it is false.

ReportWriter uses a simple IF structure to assign one of two values to the Conditional field depending on the condition. For example, you may want to determine the tax for an order. The tax depends on a condition—is the customer taxable or non-taxable? The resulting control structure would be:

```
IF CUS:Taxable                    ! conditional expression
  TAX = ORD:Total * CUS:TaxRate   ! True assignment expression
ELSE
  TAX = 0                         ! False assignment expression
END
```

The Conditional Field is defined as shown in the illustration below.



To create a Conditional Field:

1. Choose **Report** ▸ **File Schematic** (or press the  button on the toolbar).

   The *File Schematic* window appears. The File Schematic lists all of the report's files, the fields in those files, and the report-specific fields.

2. Highlight *Conditional Fields* in the File Schematic, then press the **Add Conditional Field** button.

   The *Conditional Field* window appears. This is where the field is defined.

3. Type a name for the field in the **Label** field.

   The Label can contain only alphanumeric characters (letters or numbers) or the underscore (_) character, and must begin with a letter. Blank spaces are not allowed. The maximum length is 40 characters. Each field name must be unique. There can be no duplication of names within a report.

4. Optionally, provide a **Description**.

   A description can help you identify the Conditional field.

5. Type the picture for the field in the **Picture** field, or press the ellipsis (...) button to use the Picture Editor.

   This specifies how the total field's value displays. See the *Print and Display Formatting* appendix for more information on Pictures.

6. Type an expression for the **If** Condition, or press the **Formula** button to use the Formula Editor.

   The Formula Editor lets you create a syntactically correct expression. In the Formula Editor, you add components one-by-one from a list of the available components. The Formula Editor offers only valid options every step of the way.

   Since a Conditional Field evaluates the truth of a statement, the Formula Editor creates a Conditional expression. A conditional expression evaluates as true or false. For example, the expression: Date < 06/07/1996 evaluates as true if the value of the Date field is 06/06/1996 or less. If it is greater than or equal to 06/07/1996, then the expression evaluates as false.

7. Type an expression for the **Then** Condition, or press the **Formula** button to use the Formula Editor.

   The **Then** expression's value is assigned to the field when the If condition is true.

8. Type an expression for the **Else** Condition, or press the **Formula** button to use the Formula Editor.

   The Else expression's value is assigned to the field when the If condition is not true.

To place a Conditional Field on a report:

1. Choose **Control** ▸ **Data Field**.

   The *File Schematic* window appears.

2. Highlight *Conditional Fields* in the File Schematic.

3. Highlight the Conditional Field in the Fields List, then press the **Select** button.

4. CLICK on the report at the location to place the field.

Creating a Runtime Field

Runtime fields have many uses. When a report has a runtime field, users receive a prompt asking for input when they run the report.

The runtime field can be a component in the expressions used by detail filters, record filters, computed fields, or conditional fields. A runtime field can also be printed on a report.

To create a runtime field:

1. Choose **Report** ▸ **File Schematic** (or press the  button on the toolbar).

   The File Schematic window appears. The File Schematic lists all of the report's files, the fields in those files, and the report-specific fields.

2. Highlight *Runtime* in the File Schematic, then press the **Add Runtime Field** button.

   The *Runtime Field* window appears.

3. Type a name for the field in the **Label** field.

   The Label can contain only alphanumeric characters (letters or numbers) or the underscore (_) character, and must begin with a letter. Blank spaces are not allowed. The maximum length is 40 characters. Each field name must be unique. There can be no duplicate names within a report.

4. Optionally, provide a **Description**.

   A description can help you identify the Runtime field.

5. Type the picture for the field in the **Picture** field, or press the ellipsis (...) button to use the Picture Editor.

   This specifies how the runtime field's value displays or prints. This also limits the number of characters the user can type in at runtime. See the *Print and Display Formatting* appendix for more information on Pictures.

6. Type the message that asks the user for data into **User Prompt** field.

   This is what the user sees in the window that appears before the report runs. All characters are allowed, including spaces, but the maximum is 20 characters.

7. If you want Validity checking for the Runtime Field, press the **Validation Formula** button to create the expression with the Formula Editor (see Using the Formula Editor).

   Validity Checking establishes constraints for data entry. This ensures that only valid data is in a runtime field. For example, you may want a value to be greater than ten (10). Creating this Validity Check prevents users from entering a value less than ten at runtime.

8. When you use Validation, optionally provide an **Error Message**.

   This is the message a user sees when entering invalid data in the runtime field.

9. Press the **OK** button.

The Formula Editor

The Formula Editor helps you to quickly generate a syntactically correct expression. You can use the Formula Editor to create computed fields or conditional fields.

- A computed field receives the value of an expression. In other words, a computed field is the receiving end of a simple assignment statement: variable = expression. For example, a computed field called GrossPrice might receive the result of adding two fields called BasePrice and Tax.

  You can use a computed field wherever the report needs the result of a calculation.

- A conditional field is a computed field with multiple possible assignments. The assignment statement executed depends on the evaluation of the IF condition. For example, a conditional field called “Tax” could equal 0 when “Taxable” (the IF condition) is false, or “Tax” could equal Price times TaxRate when “Taxable” is true.

  You can use a conditional field wherever the report needs different calculations based on a condition.

The *Formula Editor* dialog gives you access to data dictionary fields, as well as runtime fields and report specific fields, and facilitates creating syntactically correct expressions. This is its prime advantage: automatic syntax checking.

To create an expression, you press buttons to add expression components to the **Statement** line. You can also type in your expression and check the syntax after.

Creating an Expression

From any area where a Formula is used:

1. Press the **Formula** button.
2. In the **Statement** field, create your formula.

   You can type in the expression, use the Formula Editor’s buttons, or both. The first component of an expression must be an operand, a left parenthesis, or a unary minus (the negative sign). For example, press the **Data** button and choose a variable, press the **Multiply by** (*) button, then press the **Functions** button to choose a Clarion built-in function.
3. Press the **Check** button to check the syntax of the expression.

   If the syntax is correct, a large green check mark appears to the left of the statement. If the syntax is incorrect, a large red X appears.
4. Press the **OK** button.

Expression Components

An expression is made up of two types of components: operands and operators. Operators perform an operation (such as addition, subtraction, etc.) on one or more Operands. Operands either contain or return a value. Constants, data dictionary fields, memory variables, and functions are examples of operands. An operand can be made up of more than one component, for example, a function and its parameters.

The Formula Editor lets you choose operators and operands, then insert them into the **Statement** line.

The table below lists all the components used in expressions.

Math Operators

+	Plus sign: Adds two operands together.
-	Minus sign: Subtracts one operand from another.
*	Asterisk: Multiplies one operand by another.
/	Slash: Divides one operand by another.
%	Percent sign: Returns the remainder from a division operation (modulus division).
&	Ampersand: Appends one text string to another.
^	Caret: Raises one operand to the power of the other (exponentiation).
( )	Parentheses: Groups components together within an expression.

Logical Operators

=	Equal: Evaluates whether one expression is equal to the other.
<	Less Than: Evaluates whether one expression is less than the other.
>	Greater Than: Evaluates whether one expression is greater than the other.
<>	Not Equal: Evaluates whether one expression is not equal to the other.
>=	Greater or Equal: Evaluates whether one expression is greater than or equal to the other.
<=	Less or Equal: Evaluates whether one expression is less than or equal to the other.
AND	Connects two logical expressions together. For an expression containing an AND to be true, both expressions of the AND must be true.
OR	Connects two logical expressions together. An expression containing an OR is true if either expression of the OR is true.
NOT	Reverses the evaluation of an expression.

Operands :

Data Includes data dictionary fields, and report-specific fields.

Functions
: All of the built-in functions of the Clarion programming language. These functions all perform some operation on parameters (other operands) and return a value to the expression.

Constant Text
: You can type constant text surrounded in single quotes (‘A’) on the Statement line or type the text into the Constant Text box.

Constant Number
: You can type constant numbers on the Statement line or type the number into the Constant Number box.

Constant Date
: Lets you specify a constant date.

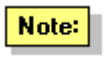
Constant Time
: Lets you specify a constant time.

10 - ADVANCED TOPICS

Using the Runtime Print Engine

Clarion ReportWriter Report Library files (.TXR) are redistributable along with the print engine (C60PRNTX.EXE) and its supporting DLLs. This allows you to distribute the files needed for end users to print or preview reports. It does not allow any modifications to reports.

To design reports, each user must own a separate copy of Clarion ReportWriter for Windows. You can purchase ReportWriter from SoftVelocity for resale to your clients or to include with your application as part of a complete solution. This allows your customers to modify your reports or create new ones.



ReportWriter (C60RW.EXE) is not redistributable. Each user must purchase a separate copy.

See *Distributing your Reports* for details on deploying Report Libraries and the necessary supporting files.

The print engine (C60PRNTX.exe) is distributable. This lets you create reports and distribute Report Libraries (.TXR files) along with the print engine to multiple users. These users will be able to run reports, preview them, and print them. However, they will not be able to modify the reports.

You can pass information to the print engine on the command line or through an Initialization (.INI) file. This tells the print engine which Report Library to use, which report to execute, plus any desired options.

Print Engine Command Line Parameters

C60PRNTX [|ReportLibraryFile |] [ReportLabel] [optionlist]
|INIFilename |

C60PRNTX The 32-bit print engine—C60PRNTX.EXE.

ReportLibraryFile The full path and filename of the Report Library File (.TXR).

IniFilename The Initialization file containing all the information to pass to the print engine.

ReportLabel The label of the report within the Report Library. If omitted, a simple list box displayed with list of available reports.

Optionlist A list of optional parameters for the print engine.

C60PRNTX.EXE are stand-alone executable files. You can install either file on any computer to execute reports. You can create Shortcuts to run from the Desktop.

You can use the print engine in many ways to run reports. By associating .TXR files with C60PRNTX.EXE, you can enable Windows Explorer double-click functionality. For example, if C60PRNTX.EXE is associated with .TXR files, double-clicking on a Report Library file in Explorer opens the Report Library with the print engine and displays a list of reports. This also allows drag and drop functionality. By dragging a .TXR file onto a print engine icon or shortcut, you can open the report library and display a list of its reports.

You can also provide optional parameters on the command line or through an Initialization file.

/Pstring The report's password. For example: /P"bat123" specifies the password is bat123.

/W[n] Preview (n pages). If n is omitted, all pages will appear. For example: /W specifies to run in Preview mode. Another example: /W4 specifies to run in Preview Mode and only show the first four pages.

/Dprinterdef The Printer Name (device) as specified in the Windows Print Manager. For example /D"HP LaserJet 4/4M PostScript" specifies the HP LaserJet printer.

/Rs-*e* The report's page range where s is the Start page and e is the End page. For example: /R6-10 specifies to print pages 6 through 10 inclusive.

/C*n* The number of copies to print is n. For example: /C6 specifies to print 6 copies of the report.

/D? Specifies to display the standard printer dialog allowing users to select the target printer and/or the range of pages.

/D?"*caption text*"
Specifies to display the standard printer dialog allowing users to select the target printer and/or the range of pages displaying the caption text on the title bar.

/Vvarname=value

This assigns values to runtime variables. There cannot be spaces on either side of the equal sign (=). You can specify more than one matched pair on the command line. For example: /Vx=3 or /Vproduct="Widgit".

/Iininame The Initialization file to use.

Examples:

```
C60PRNTX

C60PRNTX ORDERS.TXR

C60PRNTX C:\REPORTS\ORDERS.TXR

C60PRNTX ORDERS.TXR Invoice

C60PRNTX ORDERS.TXR SalesReport /VCustNumber=23

C60PRNTX ORDERS.TXR MonthlySalesReport /P"theboss"

C60PRNTX /Iinvoice.ini

C60PRNTX /Iinvoice.ini SalesReport /VCustNumber=23
```

Using C60PRNTX with an Initialization File

The Initialization file is either (in order of priority):

- The Initialization file set by the /I command line parameter.
- The Initialization file with the same name as the Report Library being printed.
- C60PRNTX.INI found by the Windows .INI search path.

In the Initialization file, there are three sections: [Run], [Variables], and [Paths]. These contain values for the command line options. Any values specified on the command line override INI file values.

INI File Format:

```
[Run]
LIBRARY=libname            ! same as 1st parameter
REPORT=repname             ! same as 2nd parameter
PASSWORD=password          ! same as /P
PREVIEW=n                  ! same as /Wn (if omitted all pages are previewed)
DEVICE=device              ! same as /D
RANGE=s-e                  ! same as /R
COPIES=n                   ! same as /C
DEVICE=?                   ! same as /D?
DEVICE=?"Caption Text"     ! same as /D?"Caption Text"
[Variables]
variablename=value         ! same as /Vvariablename=value
[Paths]
mask=dirpath
```

INI File Example:

```
[Run]
LIBRARY=C:\C60\EXAMPLES\RWTUTOR\RWTUTOR.TXR
REPORT=StudentLabels
PASSWORD=SoftVelocity
PREVIEW=5
RANGE=6-10
COPIES=6
DEVICE=?"Specify Target Printer"
[Variables]
MajorCode=5
[Paths]
*.TPS=C:\C60\RWTUTOR
```

Distributing your Reports

To distribute reports to computers which do not have ReportWriter installed, you must provide the following files:

Your Report Library file	(*name*.TXR)
The runtime Print Engine	C60PRNTX.EXE
The Runtime Library file:	C60RUNX.DLL

Any Data Files the report uses.

File Driver(s):

The appropriate File Driver for the format of your data files (one or more from the list below):

File	Library
ASCII	C60ASCX.DLL
BASIC	C60BASX.DLL
Btrieve	C60BTRX.DLL
Clarion	C60CLAX.DLL
Clipper	C60CLPX.DLL
dBase III	C60DB3X.DLL
dBase IV	C60DB4X.DLL
DOS	C60DOSX.DLL
FoxPro	C60FOXX.DLL
ODBC	C60ODBX.DLL
Scalable	C60SCAX.DLL
TopSpeed	C60TPSX.DLL

Install the Print engine and the supporting files in a single directory. The Report Library and data files can be in the same directory or any other directory. You can also use search paths to “point” the library to the location of the data files.

Optionally, you can set up Program Manager Icons or Shortcuts to run individual reports.

11 - PROGRAMMER'S GUIDE TO REPORTWRITER

This chapter documents methods Clarion programmers can use to extend ReportWriter's capabilities, including how to incorporate ReportWriter's print engine into an application and techniques to add external libraries to ReportWriter. This chapter is intended only for Clarion programmers and the techniques require the Clarion development environment.

If you are an end user of ReportWriter but not a programmer, you can benefit by reading this chapter only if you intend to program with Clarion or hire a Clarion developer. If you have purchased a copy of ReportWriter to use with a program developed with Clarion, you can contact that developer to request custom libraries to add the functionality you want. Reading this chapter, may help you determine what is possible or give you some ideas to discuss with the programmer.

Adding External Libraries to a Report Library

Using Clarion, you can create external libraries to extend ReportWriter's functionality. This allows you to add user defined functions and variables to formula fields used in reports. There are three steps to extending ReportWirter:

- Create the DLL containing a function or variable you want to use in a report.
- Register the external Library in a Report Library.
- Call the external function in a computed or conditional field.

Creating External DLLs to use in ReportWriter

To create the DLL, follow the instructions in the Clarion Programmer's Guide. There are, however, some additional requirements for creating DLLs to use with ReportWriter.

Prototypes

Prototype all functions you want to use with ReportWriter to accept STRING parameters only.

Name

In order to reference an exported function, you must know its exported name. This name is never the same as the function itself, unless you provide a NAME attribute. Using a NAME attribute, you ensure that the name you are referencing is the same as the function name. If you create your DLL in Application Generator, the export file (.EXP) is automatically created for you.

Return Data Types

Exported functions you want to use with ReportWriter should only return STRING data. Exported variables must be BINDed and be STRING data types.

Deployment of external DLLs for ReportWriter

External variables and functions are available to all reports in the report library and thus only need to be defined once for each library.

Once defined, the external variables and functions can be incorporated into any formula field (such as a computed field) by selecting the appropriate external resource from the Externals list in the formula editor. The description displayed in the formula editor tree is the User Description entered when defining the external and can contain parameter descriptions for easy reference.

During report printing, functions will be called every time a formula that uses that function is used.

To use an external resource, you must "register" it in a Report Library.

From the Report Library window:

1. Choose **File** ▸ **Externals**.

From the Report Formatter:

1. Choose **Report** ▸ **Externals**.

   The *External Procedures and Variables* dialog appears.

2. To add an external resource, press the **New External** button. To change the specifications, press the **Edit External** button.

   The *External* dialog appears. This is where you tell ReportWriter how it will access the external function or variable.

3. Complete the information on the External dialog.

   Label
   The name of the function or variable to use in the formula.

   Description
   A "friendly" description to display in the file schematic.

   Function
   If the external resource is a function, check this box. Clear the box if the external resource is a variable.

   Number of parameters (for functions)
   If the external resource is a function, specify the number of parameters (up to 16 supported). This must exactly match the prototype in the DLL.

   String size (for variables)
   If the external is a variable, specify the number of characters in the variable. This must exactly match the variable's definition in the DLL.

DLL name
This specifies the DLL where the function or variable is implemented. Only the filename of the DLL should be specified (without the extension). If this is left blank, it is assumed the external will be bound by a program calling C60PRLBX. An invalid formula error will be displayed if an 'unbound' external is used.

DLL Entrypoint
Specifies the name of the function or variable as exported. This must match the name in the EXP file when creating the external DLL. Names must be exported all uppercase.

Integrating the Report Engine into Clarion Applications

The ReportWriter Report Engine is supplied in two versions:

Executable	C60PRNTX.EXE
Dynamic Link Library	C60PRLBX.DLL

The Report Engine is easily driven programmatically as a Dynamic Link Library (C60PRLBX.DLL). It contains a simple object oriented interface that enables:

- Report libraries (i.e. TXR files) to be read and reports printed.
- The report preview to be replaced by user code.
- The record fetch mechanism to be tailored to user requirements.

There is one class, ReportEngine, defined in RWPRLIB.INC which handles all the above functionality. To show how easy it can be to print a report from a Clarion program, the following is a full program that prints Report1 from RWDEMO1.TXR.

```
  PROGRAM
  MAP
  END
  INCLUDE('RWPRLIB.INC')
RE   ReportEngine                      ! declare ReportEngine object
   CODE
   RE.LoadReportLibrary('RWDEMO1.TXR')  ! load report library
   RE.PrintReport('Report1')            ! print report
```

ReportEngine Methods

The Report Engine class has built-in methods you can use to control the manner in which reports print. You instantiate a ReportEngine object in your code, then call any of the methods.

In addition, this also provides the ability to derive and override or enhance any of these methods. For example, you could define a method in your object and add additional functionality by calling the PARENT method from your method.

All the prototypes for the Report Engine Class are in RWPRLIB.INC.

The Report Engine class contains the following methods:

LoadReportLibrary (Load a ReportWriter Report Library)

LoadReportLibrary(libraryname [, password]),PROC,VIRTUAL

LoadReportLibrary	Load a report library (.TXR) file.
libraryname	The filename of the report library containing the report(s) to be printed.
password	The report library's password (if needed). If omitted, no password is passed to the report library.

LoadReportLibrary loads a report library (.TXR) file. This must be called before attempting to set any properties or printing a report. Only one ReportLibrary can be loaded for each ReportEngine object (loading a TXR unloads any TXRs previously loaded within that object).

This method returns TRUE if the specified TXR loads with no errors.

Return Data Type: SIGNED

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')   ! first load the report
    IF NOT RE.PrintReport('Report1')       ! run preview/print (report1)
      MESSAGE('Print Failed')
    END
    RE.UnloadReportLibrary
  ELSE
    MESSAGE('Load Failed')
  END
```

See Also: UnloadReportLibrary

PrintReport (print a report)

PrintReport (*reportname* **),PROC,VIRTUAL**

PrintReport Print a report in the current report library (.TXR) file.

reportname The name or number of the report to print. If an empty string (''), a list of available reports is displayed allowing the user to select a report.

PrintReport prints the specified report in the current report library (.TXR) file. The report can go directly to the printer or to a report preview (see SetPreview).

This method returns TRUE if there were no errors during printing.

Return Data Type: SIGNED

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')   ! first load the report
    IF NOT RE.PrintReport('Report1')       ! run preview/print (report1)
      MESSAGE('Print Failed')
    END
    RE.UnloadReportLibrary
  ELSE
    MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, SetPreview

UnloadReportLibrary (Unload a ReportWriter Report Library)

UnloadReportLibrary ,VIRTUAL

UnloadReportLibrary unloads the current report library (.TXR) file. This frees resources allocated to a report library and clears any parameters set. It generally does not need to be called as it is automatically invoked by either loading a new report library or when the ReportEngine object is freed.

This method returns TRUE if the specified TXR loads with no errors.

Return Data Type: SIGNED

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')   ! first load the report
    IF NOT RE.PrintReport('Report1')       ! run preview/print (report1)
      MESSAGE('Print Failed')
    END
    RE.UnloadReportLibrary
  ELSE
    MESSAGE('Load Failed')
  END
```

See Also: UnloadReportLibrary

SetVariable (Set a value to a runtine variable)

SetVariable(*variablename* **,** *value* **),VIRTUAL**

SetVariable Sets a value to a runtime variable in a report.

variablename The label of the runtime variable in the report.

value The value to set to the runtime variable.

SetVariable sets a value to a runtime variable in a report.

LoadReportLibrary must be called before attempting to use this method. The SetVariable method should be called to set a variable's value before calling the PrintReport method.

The values persist until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')! first load the report
   RE.SetVariable('UserName','Jimmy')   ! after library loaded/before printing
   IF NOT RE.PrintReport('Report1')     ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
   MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary

SetPreview (Set report's preview on or off)

SetPreview([*numberofpages* **]),VIRTUAL**

SetPreview	Sets the report's preview mode.
numberofpages	The number of pages to preview. If omitted, all pages are previewed.

SetPreview sets the report's preview mode. This defines whether a preview dialog is displayed before printing.

If *numberofpages* is omitted (or negative) then preview is enabled for all pages of report. If *numberofpages* is 0, preview is disabled. This is the default after loading a report library. If *numberofpages* is positive only that number of pages display in the preview (although all pages will print).

LoadReportLibrary must be called before attempting to use this method. The SetPreview method should be called before calling the PrintReport method.

The value persists until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')! first load the report
   RE.SetPreview(5)                     ! after library loaded/before printing
   IF NOT RE.PrintReport('Report1')     ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
   MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary

SetPrinter (Set printer for report)

SetPrinter(*printerdef* **),VIRTUAL**

SetPrinter Sets the report's printer.

printerdef The printer name (device) as defined in the Windows Print Manager.

SetPrinter sets the report's printer. The *printerdef* must match a definition in the Windows Print Manager.

LoadReportLibrary must be called before attempting to use this method. The SetPrinter method should be called before calling the PrintReport method.

The value persists until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr') ! first load the report
   RE.SetPrinter('hp4mv')                ! after library loaded/before printing
   IF NOT RE.PrintReport('Report1')      ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
   MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary

SetPageRange (Set range of pages for report)

SetPageRange(*frompage* [, *topage*] **),VIRTUAL**

SetPageRange	Sets the range of pages for a report.
frompage	The first page to include.
topage	The last page to include.

SetPageRange sets the range of pages for a report. The *frompage* defines the beginning page of the report and the *topage* defines the ending page of the report. If the *topage* is omitted the report prints to the end. If this method is not called all pages print (the default).

LoadReportLibrary must be called before attempting to use this method. The SetPageRange method should be called before calling the PrintReport method.

The value persists until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')! first load the report
   RE.SetPageRange(33,70)               ! after library loaded/before printing
   IF NOT RE.PrintReport('Report1')     ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
    MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary

SetNumberOfCopies (Set number of copies to print for report)

SetNumberOfCopies (*numcopies* **),VIRTUAL**

SetNumberOfCopies Sets the number of copies to print for a report.

numcopies The number of copies.

SetNumberOfCopies sets the number of copies (*numcopies*) to print for a report. If this method is not called, one copy prints (the default). This may not be supported on all printers.

LoadReportLibrary must be called before attempting to use this method. The SetNumberOfCopies method should be called before calling the PrintReport method.

The value persists until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr') ! first load the report
   RE.SetNumberOfCopies(3)               ! after library loaded/before printing
   IF NOT RE.PrintReport('Report1')      ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
   MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary

SetNextPageNumber (Set the next page number for report)

SetNextPageNumber (*pagenum* **),VIRTUAL**

SetNextPageNumber Sets the number of the next page of a report.

Pagenum The number of the next page.

SetNextPageNumber sets the number (*pagenum*) of the next page of a report. This defines the value printed for the first and following (sequentially increasing) page numbers. This allows to reports to be printed using sequential page numbering

LoadReportLibrary must be called before attempting to use this method. The SetNextPageNumber method should be called before calling the PrintReport method.

The value persists until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr') ! first load the report
   RE.SetNextPageNumber(23)              ! after library loaded/before printing
   IF NOT RE.PrintReport('Report1')      ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
    MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary, GetNextPageNumber

GetNextPageNumber (Get the next page number for report)

GetNextPageNumber,VIRTUAL

GetNextPageNumber Gets the number of the next page of a report.

pagenum The number of the next page.

GetNextPageNumber retrieves the number (*pagenum*) of the next page that would print after the report. In other words one more than the last page of a report. This is useful when used in conjunction with SetNextPageNumber by allowing you access to the page number to use for the starting number of the next report.

LoadReportLibrary must be called before attempting to use this method. The GetNextPageNumber method should be called after calling the PrintReport method.

Return Data Type: LONG

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')   ! first load the report
   IF NOT RE.PrintReport('Report1')     ! run preview/print (report1)
    MESSAGE('Print Failed')
   ELSE
    RE.SetNextPageNumber(RE.GetNextPageNumber())
    IF NOT RE.PrintReport('Report2')     ! run preview/print (report1)
     MESSAGE('Print Failed')
    END
   END
   RE.UnloadReportLibrary
  ELSE
    MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary, SetNextPageNumber

SetReportFilter (Set a report's filter)

SetReportFilter (*reportname, filter* **) ,VIRTUAL**

SetReportFilter	Overrides the report's filter.
reportname	The name of the report.
filter	The new value for the filter.

SetReportFilter overrides the report's filter, setting it to the specified filter.

LoadReportLibrary must be called before attempting to use this method. The SetReportFilter method should be called before calling the PrintReport method.

The value persists until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')   ! first load the report
   RE.SetReportFilter('Report1','demo:Major<<=3')
   IF NOT RE.PrintReport('Report1')        ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
    MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary

SetReportOrder (Set a report's order)

SetReportOrder(*reportname*, *order***) ,VIRTUAL**

SetReportOrder	Overrides the report's sort order.
reportname	The name of the report.
order	The new sort order for the report.

SetReportOrder overrides the report's sort order, setting it to the specified order. See the Language Reference for a more information about the order string format.

LoadReportLibrary must be called before attempting to use this method. The SetReportOrder method should be called before calling the PrintReport method.

The value persists until a new report library is loaded or the UnloadReportLibrary method is called.

Example:

```
 PROGRAM
 MAP
 END
 INCLUDE('RWPRLIB.INC')

RE ReportEngine

  CODE
  IF RE.LoadReportLibrary('rwdemo1.txr')   ! first load the report
   RE.SetReportOrder('Report1','UPPER(STU:FirstName)')
   IF NOT RE.PrintReport('Report1')     ! run preview/print (report1)
    MESSAGE('Print Failed')
   END
   RE.UnloadReportLibrary
  ELSE
   MESSAGE('Load Failed')
  END
```

See Also: LoadReportLibrary, PrintReport, UnloadReportLibrary

Advanced Techniques

The following techniques are for more advanced control when printing reports. The Report Engine allows you to "take over" some of its internal functions to allow you to control it in any way needed. This is achieved by 'taking over' the virtual PrintHook for your own class.

```
MyEngine CLASS(ReportEngine)
PrintHook           PROCEDURE(),SIGNED,VIRTUAL
         END
```

PrintHook is called from within the PrintReport method after the report definition is created. It is the PrintHook method's job to read the data from the file(s) and then print the details to the report. It does this by calling the following three helper functions:

```
Reset               PROCEDURE(),LONG,VIRTUAL   ! returns records to process
Next                PROCEDURE(),BYTE,VIRTUAL   ! returns Level:Notify at end
PrintAction         PROCEDURE(),VIRTUAL        ! called after next
EndReport           PROCEDURE(),VIRTUAL        ! call to close report
```

and using the following properties of the ReportEngine class:

```
ReportName          STRING(FILE:MaxFilePath)
Report              &WINDOW
View                &VIEW
PagesToPreview      LONG           ! 0  = no preview
                                   ! -1 = all pages
```

These methods and properties are only available during the PrintHook call. They are cannot be validly used at any other time.

The simplest printhook implementation (without preview) would be:

```
MyEngine.PrintHook           PROCEDURE(),SIGNED,VIRTUAL
  CODE
    LOOP
      SELF.Reset()                          ! Reset files/view
      LOOP
        CASE SELF.Next()                    ! Record fetch
        OF Level:Notify
          RETURN TRUE
        OF Level:Fatal
          RETURN FALSE
        END
        SELF.PrintAction()                  ! Print detail(s)
      END
    END
    SELF.EndReport()                        ! Close the report
```

This cycles through the records calling PrintAction for every record fetch.

To include print preview, the code is considerably more complex but you can easily use ABC Library (ABBROWSE and ABREPORT).

See **RWDEMO2.CLW** for a model implementation including the handling of the progress window.

Replacing Record Fetch

For the advanced control of reports, you can derive the ReportEngine class and override the record fetch helper routines Reset and Next. This powerful engine allows for database access that cannot be specified using views. By using bound functions and variables it is possible to divorce the report from any database access entirely.

Reset must initialize the file access so that the subsequent call to *Next* will return the first record that is to printed. It should return the number of records to print (if known) or -1 if not known or it is unreasonable to calculate.

Next must fetch the next record to make it available for printing (via PrintAction). It must return Level:Benign if succesful. Level:Notify if no more records are available or Level:Fatal if there has been an error.

For example:

```
 PROGRAM
 MAP
   StopOnError
 END
 INCLUDE('RWPRLIB.INC')
 INCLUDE('ABERROR.INC')

Q QUEUE,PRE(Q),BINDABLE
Title  STRING(40),NAME('BookTitle')
Author STRING(30),NAME('BookAuthor')
  END

QPos SHORT

RE class(ReportEngine)
Reset               PROCEDURE(),LONG,VIRTUAL ! returns records to process
Next                PROCEDURE(),BYTE,VIRTUAL ! returns 0 if finished
  END

  CODE
    Q:Title = 'Valis'
    Q:Author = 'Philip K. Dick'
    ADD(Q)
    Q:Title = 'At Swim Two Birds'
    Q:Author = 'Flann O''Brien'
    ADD(Q)
    Q:Title = 'Roads to Freedom'
    Q:Author = 'John Paul Sartre'
    ADD(Q)
    Q:Title = 'The Painted Bird'
    Q:Author = 'Jerzy Kosinski'
    ADD(Q)
    Q:Title = 'Staring at the Sun'
    Q:Author = 'Julian Barnes'
    ADD(Q)
```

```
    BIND(Q)
    RE.LoadReportLibrary('rwdemo2.txr') ! Load report library
    RE.SetPreview()                     ! enable preview
    RE.PrintReport('Report2')           ! Print report
    StopOnError
    RE.UnloadReportLibrary
    StopOnError

StopOnError PROCEDURE
  CODE
  IF ERRORCODE() = 90 THEN
    STOP(CLIP(FILEERROR()) & ' (' & CLIP(FILEERRORCODE()) & ')')
  ELSIF ERRORCODE()<>0 THEN
    STOP(CLIP(ERROR()) & ' (' & ERRORCODE() & ')')
  END

RE.Reset PROCEDURE
  CODE
  QPos = 0
  RETURN RECORDS(Q)

RE.Next  PROCEDURE
  CODE
  QPos += 1
  IF QPos>RECORDS(Q) THEN
    RETURN Level:Notify
  END
  GET(Q,QPos)
  RETURN Level:Benign
```

Other Micellaneous Hooks

These additional hooks into the Print Engine enable additional control over your reports.

AttachOpenFile (Attach an open file to a report)

AttachOpenFile(*label***) ,VIRTUAL**

AttachOpenFile Attaches an open file to the report and overrides the report's file.

label The label of the file.

AttachOpenFile attaches an open file to the report and override the report's file. This allows you to replace the file specified in the report library with an open file. The label passed is the name of the file to replace. The function should return the FILE if it is required to be replaced, or should 'pass-on' the call to PARENT.AttachOpenFile if not.

This function need only be called in when the file to print is different from the file specified in the report library. You must ensure the FILE returned is already OPEN. The print engine does not open it. If the file definition is different to the original file specified in the report, then the all field names used must match in name and prefix. If the name and prefix do not match then you will need to BIND the fields used.

Return Data Type: FILE

Example:

```
RE CLASS(ReportEngine)
AttachOpenFile    PROCEDURE(STRING label),*FILE,VIRTUAL ! To attach my own file
   END

RE.AttachOpenFile    PROCEDURE(STRING label)
  CODE
    IF label='students' THEN                ! file to override
      RETURN students2                       ! new file to attach
    END
    RETURN Parent.AttachOpenFile(label)     ! leave others default
```

ResolveVariableFilename (Specify value for a variable filename)

ResolveVariableFilename (*vname, value***) ,VIRTUAL**

ResolveVariableFilename Resolves a variable filename used in a report.

vname The label of the variable.

value The value to supply for the variable.

ResolveVariableFilename resolves a variable filename used in a report. This can be used for filenames that were specified to be variables in the dictionary used to create a report library. Using this function, you can avoid the dialog prompting for the variable name. This function should return TRUE if the variable name is recognized and value set to the filename, otherwise it should return FALSE.

Return Data Type: SIGNED

Example:

```
RE  CLASS(ReportEngine)
ResolveVariableFilename PROCEDURE(STRING vname,*STRING value),SIGNED,VIRTUAL
    END

RE.ResolveVariableFilename PROCEDURE(STRING vname,*STRING value)
  CODE
    IF label='!avariable'
      value = 'c:\examples\test.tps'
      RETURN TRUE
    END
    RETURN FALSE
```

ReadReportLibrary (Read a Report Library into buffer)

ResolveVariableFilename (*buffer, count***) ,VIRTUAL**

ReadReportLibrary Reads a Report Library into buffer.

buffer The buffer to hold the report library read.

count The value to supply for the variable.

ReadReportLibrary reads a Report Library into the buffer. This allows you to load a report library from a data element instead of a .TXR file (e.g., from a database). The function will be called by LoadReportLibrary to read the library in 64 KB chunks. A derived ReadReportLibrary should copy the contents of the report library into the supplied buffer and return the amount read. When the library has been fully read the function should return 0.

Return Data Type: SHORT

Example:

```
 PROGRAM
 MAP
 END
INCLUDE('RWPRLIB.INC')

NL EQUATE('<13,10>')

TestReport STRING( |
'[REPORTS]'                                                          & NL & |
'Report1 REPORT,FONT(''Arial'',10),PRE(Report1),THOUS,AT(1000,1662,6500,8338)'&NL&|
'_REPORT_  BREAK'                                                     & NL & |
'_TOTALS_    BREAK'                                                   & NL & |
'Detail1       DETAIL,FONT(''Arial'',8,0),AT(0,0,,1031)'              & NL & |
'                STRING(@s20),USE(stu:FirstName),AT(51,0)'            & NL & |
'                STRING(@s20),USE(stu:LastName),AT(1565,0)'           & NL & |
'                STRING(@s12),USE(stu:Telephone),AT(3079,0)'          & NL & |
'              END'                                                   & NL & |
'            END'                                                     & NL & |
'          END'                                                       & NL & |
'          FOOTER,AT(1000,9000,,1000)'                                & NL & |
'            STRING(@n5),PAGENO,AT(813,396,323)'                      & NL & |
'            STRING(''Page:''),AT(396,396)'                           & NL & |
'          END'                                                       & NL & |
'        END'                                                         & NL & |
'[FILES]'                                                            & NL & |
'students FILE,PRE(stu),DRIVER(''TOPSPEED''),NAME(''students.tps'')'  & NL & |
'KeyStudentNumber KEY(stu:Number),NOCASE,OPT'                        & NL & |
'MajorKey  KEY(stu:Major,stu:LastName,stu:FirstName),DUP,NOCASE,OPT'  & NL & |
'KeyLastName KEY(stu:LastName),DUP,NOCASE'                           & NL & |
'KeyGradYear KEY(-stu:GradYear,stu:LastName,stu:FirstName),DUP,NOCASE,OPT' & NL & |
'Photo     MEMO(64000),BINARY'                                       & NL & |
```

```
'__Record  RECORD'                                                    & NL & |
'Number      LONG'                                                    & NL & |
'FirstName   STRING(20)'                                              & NL & |
'LastName    STRING(20)'                                              & NL & |
'Address     STRING(20)'                                              & NL & |
'Address2    STRING(20)'                                              & NL & |
'City        STRING(20)'                                              & NL & |
'State       STRING(2)'                                               & NL & |
'Zip         LONG'                                                    & NL & |
'Telephone   STRING(12)'                                              & NL & |
'Major       LONG'                                                    & NL & |
'GradYear    LONG'                                                    & NL & |
'          END'                                                       & NL & |
'        END'                                                         & NL & |
                                                                      NL & |
'[REPORTVIEWS]'                                                       & NL & |
'Report1 VIEW(students),ORDER(''UPPER(stu:LastName)''),KEY(stu:KeyLastName)'&NL& |
'        END'                                                         & NL )

RE CLASS(ReportEngine)
ReadReportLibrary   PROCEDURE(*CSTRING buffer,USHORT count),SHORT,VIRTUAL
Pos                 SHORT
  END

  CODE
    RE.Pos = 0
    RE.LoadReportLibrary('')
    RE.SetPreview()
    RE.SetNextPageNumber(10)
    RE.PrintReport('Report1')
    RE.UnloadReportLibrary

RE.ReadReportLibrary   PROCEDURE(*CSTRING buffer,USHORT count)
! return report definition
cntr USHORT
i SHORT
  CODE
    cntr = SIZE(TestReport)-SELF.Pos
    IF cntr>count THEN cntr=count.
    LOOP i = 1 TO cntr
      SELF.Pos += 1
      buffer[i-1] = TestReport[SELF.Pos]
    END
    RETURN cntr            !will return zero at end
```

APPENDIX A - PRINT AND DISPLAY FORMATTING

Picture Tokens

ReportWriter uses picture tokens to provide a masking format for printing and displaying variables. Picture tokens define the way a field prints on a report.

There are seven types of picture tokens:

Numeric and currency

Scientific notation

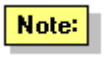
Date

Time

Pattern

Key-in template

String



Key-in Template picture tokens are used for data entry only, so they only apply to Runtime Fields.

Numeric and Currency Pictures

@**N** **[***currency***]** **[***sign***]** **[***fill***]** *size* **[***grouping***]** **[***places***]** **[***sign***]** **[***currency***]** **[***B***]**

@N All numeric and currency pictures begin with @N.

currency Either a dollar sign ($) or a string constant enclosed in tildes (~). When it precedes the sign indicator and there is no fill indicator, the currency symbol "floats" to the left of the high order digit. If there is a fill indicator, the currency symbol remains fixed in the left-most position. If the currency indicator follows the size and grouping, it appears at the end of the number displayed.

sign Specifies the display format for negative numbers. If a hyphen precedes the fill and size indicators, negative numbers will display with a leading minus sign. If a hyphen follows the size, grouping, places, and currency indicators, negative numbers will display with a trailing minus sign. If parentheses are placed in both positions, negative numbers will be displayed enclosed in parentheses.

fill Specifies leading zeros, spaces, or asterisks (*) in any leading zero positions and suppresses grouping. If the fill indicator is omitted, leading zeros are suppressed and grouping is enabled.

0 (zero) produces leading zeroes

_ (underscore) produces leading spaces

* (asterisk) produces leading asterisks

size The size is required to specify the total number of significant digits to display, including the number of digits in the places indicator and any formatting characters.

grouping A grouping symbol, other than a comma (the default), can be placed to the right of the size indicator to specify a three digit group separator.

. (period) produces periods

- (hyphen) produces hyphens

_ (underscore) produces spaces

places Specifies the decimal separator symbol and the number of decimal digits. The number of decimal digits must be less than the size indicator. The decimal separator may be a period (.), grave accent (‘), or the letter "v" (used only for STRING field storage declarations—not for display).

. (period) produces a period

‘ (grave accent) produces a comma

v produces no decimal separator

B Specifies that the format displays as blank whenever its value is zero. This may be upper or lower case (B or b).

The numeric and currency pictures format numeric values for screen display or in reports. If the value is greater than the maximum value the picture can display, a string of asterisks is displayed.

Example:

```
Numeric       Result        Format
@N9           4,550,000     Nine digits, group with commas (default)
@N_9B         4550000       Nine digits, no grouping, leading blanks if zero
@N09          004550000     Nine digits, leading zero
@N*9          ***45,000     Nine digits, asterisk fill, group with commas
@N9_          4 550 000     Nine digits, group with spaces
@N9.          4.550.000     Nine digits, group with periods

Decimal       Result        Format
@N9.2         4,550.75      Two decimal places, period decimal separator
@N_9.2B       4550.75       Two decimal places, period decimal separator, no
                            grouping, blank if zero
@N_9'2        4550,75       Two decimal places, comma decimal separator
@N9.'2        4.550,75      Comma decimal separator, group with periods
@N9_'2        4 550,75      Comma decimal separator, group with spaces

Signed        Result        Format
@N-9.2B       -2,347.25     Leading minus sign, blank if zero
@N9.2-        2,347.25-     Trailing minus sign
@N(10.2)      (2,347.25)    Enclosed in parens when negative

Dollar        Result        Format
@N$9.2B       $2,347.25     Leading dollar sign, blank if zero
@N$10.2-      $2,347.25-    Leading dollar sign, trailing minus when negative
@N$(11.2)     $(2,347.25)   Leading dollar sign, in parens when negative

Currency      Result        Format
@N12_'2~ F~   1 5430,50 F   France
@N~L. ~12'    L. 1.430.050  Italy
@N~£~12.2     £1,240.50     United Kingdom
@N~kr~12'2    kr1.430,50    Norway
@N~DM~12'2    DM1.430,50    Germany
@N12_'2~ mk~  1 430,50 mk   Finland
@N12'2~ kr~   1.430,50 kr   Sweden

Storage-Only Pictures:

Variable1 STRING(@N_6v2)      !Declare as 6 bytes stored without decimal
 CODE
 Variable1 = 1234.56          !Assign value, stores '123456' in file
 SHOW(1,1,Variable1,@N_7.2)   !Display with decimal point: '1234.56'
```

Scientific Notation Pictures

@Em.n[B]

@E All scientific notation pictures begin with @E.

m Determines the total number of characters in the format provided by the picture.

n Indicates the number of digits that appear to the left of the decimal point.

B Specifies that the format displays as blank when the value is zero.

The scientific notation picture formats very large or very small numbers. The format is a decimal number raised by a power of ten.

Example:

Picture	Value	Result
@E9.0	1,967,865	.20e+007
@E12.1	1,967,865	1.9679e+006
@E12.1B	0	
@E12.1	-1,967,865	-1.9679e+006
@E12.1	.000000032	3.2000e-008

Date Pictures

@Dn [s] [B] [*direction* **[***range***]]**

@D	All date pictures begin with @D.
n	Determines the date picture format. Date picture formats range from 1 through 18. A leading zero (0) indicates a zero-filled day or month.
s	A separation character between the month, day, and year components. If omitted, the slash (/) appears. . (period) Produces periods ‘ (grave accent) Produces commas - (hyphen) Produces hyphens _ (underscore) Produces spaces
B	Specifies that the format displays as blank when the value is zero.
direction	A right or left angle bracket (> or <) that specifies the “Intellidate” direction (> indicates future, < indicates past) for the range parameter. Valid only on date pictures with two-digit years.
Range	An integer constant in the range of zero (0) to ninety-nine (99) that specifies the “Intellidate” century for the direction parameter. Valid only on date pictures with two-digit years. If omitted, the default value is 80.

Dates may be stored in numeric variables (usually LONG), a DATE field (for Btrieve compatibility), or in a STRING declared with a date picture. A date stored in a numeric variable is called a “Clarion Standard Date.” The stored value is the number of days since December 28, 1800. The date picture token converts the value into one of the date formats.

The century for dates in any picture with a two-digit year is resolved using “Intellidate” logic. Date pictures that do not specify direction and range parameters assume the date falls in the range of the next 20 or previous 80 years. The direction and range parameters allow you to change this default. The direction parameter specifies whether the range specifies the future or past value. The opposite direction then receives the opposite value (100-range) so that any two-digit year results in the correct century.

For example, the picture @D1>60 specifies using the appropriate century for each year 60 years in the future and 40 years in the past. If the current year is 1996, when the user enters “5/01/40,” the date is in the year 2040, and when the user enters “5/01/60,” the date is in the year 1960.

Example:

```
Picture       Format       Result
@D1           mm/dd/yy     10/31/59
@D1>40        mm/dd/yy     10/31/59            !This defaults to 1959
@D01          mm/dd/yy     01/01/95
@D2           mm/dd/yyyy   10/31/1959
@D3           mmm dd, yyyy OCT 31,1959
@D4           mmmmmmmmm dd, yyyy  October 31, 1959
@D5           dd/mm/yy     31/10/59
@D6           dd/mm/yyyy   31/10/1959
@D7           dd mmm yy    31 OCT 59
@D8           dd mmm yyyy  31 OCT 1959
@D9           yy/mm/dd     59/10/31
@D10          yyyy/mm/dd   1959/10/31
@D11          yymmdd       591031
@D12          yyyymmdd     19591031
@D13          mm/yy        10/59
@D14          mm/yyyy      10/1959
@D15          yy/mm        59/10
@D16          yyyy/mm      1959/10
@D17                       Windows Control Panel setting for Short Date
@D18                       Windows Control Panel setting for Long Date
      Alternate separators
@D1.          mm.dd.yy     Period separator
@D2-          mm-dd-yyyy   Dash separator
@D5_          dd mm yy     Underscore produces space separator
@D6‘          dd,mm,yyyy   Grave accent produces comma separator
```

Time Pictures

@Tn[s][B]

@T All time pictures begin with @T.

n Determines the time picture format. Time picture formats range from 1 through 6.

s A separation character. Colon (:) characters appear between the hour, minute, and second components of certain time picture formats. The following s indicators provide an alternate separation character for these formats.

. (period) produces periods

‘ (grave accent) produces commas

- (hyphen) produces hyphens

_ (underscore) produces spaces

? (question mark) internationalized time

B Specifies that the format displays as blank when the value is zero.

Times may be stored in a numeric variable (usually a LONG), a TIME field (for Btrieve compatibility), or in a STRING declared with a time picture. A time stored in a numeric variable is called a “Standard Time.” The stored value is the number of hundredths of a second since midnight. The picture token converts the value to one of the six time formats.

Time pictures using the ? separation character produces a time with the separation characters specified by DOS’ Country.SYS file, based upon the country code.

Time pictures containing alphabetic characters (@T3, @T6) may not be used for data entry.

Example:

```
Picture       Format       Result
@T1           hh:mm        17:30
@T2           hhmm         1730
@T3           hh:mmXM      5:30PM
@T4           hh:mm:ss     17:30:00
@T5           hhmmss       173000
@T6           hh:mm:ssXM   5:30:00PM
@T7                        Windows Control Panel setting for short time
@T8                        Windows Control Panel setting for long time

     Alternate separators:
@T1.   hh.mm               Period separator
@T1-   hh-mm               Dash separator
@T3_   hh mmXM             Underscore produces space separator
@T4‘   hh,mm,ss            Grave accent produces comma separator
```

Pattern Pictures

@P[<][#][x]P[B]

@P All pattern pictures begin with the @P delimiter and end with the P delimiter. The case of the delimiters must be the same.

< Specifies an integer position that is blank when zero.

Specifies an integer position.

x Represents optional display characters. These characters appear in the final result string.

P All pattern pictures must end with P. If a lower case @p delimiter is used, the ending P delimiter must also be lower case.

B Specifies that the format displays as blank when the value is zero.

Pattern pictures contain optional integer positions and optional edit characters. Any character other than < or # is considered an edit character which will appear in the formatted picture string. The @P and P delimiters are case sensitive. Therefore, an upper case "P" can be included as an edit character if the delimiters are both lower case "p" and vice versa.

Pattern pictures do not recognize decimal points in order to permit the period to be used as an edit character. Therefore, the value formatted by a pattern picture should be an integer. If a floating point value is formatted by a pattern picture, only the integer portion of the number will appear in the result.

Example:

Picture	Value Entered	Result
@P###-##-####P	215846377	215-84-6377
@P<#/##/##P	103159	10/31/59
@P(###)###-####P	3057854555	(305)785-4555
@P###/###-####P	7854555	000/785-4555
@p<#:##PMp	530	5:30PM
@P<#' <#"P	506	5' 6"
@P<#lb. <#oz.P	902	9lb. 2oz.
@P4##A-#P	112	411A-2
@PA##.C#P	312.45	A31.C2

Key-in Template Pictures

@K[@] [#] [<] [x] [\] [?] [^] [_] [|]K[B]

@K All key-in template pictures begin with the @K delimiter and end with the K delimiter. The case of the delimiters must be the same.

@ Specifies only uppercase and lowercase alphabetic characters.

Specifies an integer 0 through 9.

< Specifies an integer that is blank for high order zeros.

x Represents optional constant display characters (any displayable character). These characters appear in the final result string.

**** Indicates the following character is a display character. This allows you to include any of the picture formatting characters (@,#,<,\,?,^,_,|) within the string as a display character.

? Specifies any character may be placed in this position.

^ Specifies only uppercase alphabetic characters in this position.

_ Specifies only lowercase alphabetic characters in this position.

| Allows the operator to “stop here” if there are no more characters to input. Only the data entered and any display characters up to that point will be in the string result.

K All key-in template pictures must end with K. If a lower case @k delimiter is used, the ending K delimiter must also be lower case.

B Specifies that the format displays as blank when the value is zero.

Key-in pictures may contain integer positions (# <), alphabet character positions (@ ^ _), any character positions (?), and display characters. Any character other than a formatting indicator is considered a display character, which appears in the formatted picture string. The @K and K delimiters are case sensitive. Therefore, an upper case “K” may be included as a display character if the delimiters are both lower case “k” and vice versa.

Key-in pictures are used specifically with STRING, PSTRING, and CSTRING fields to allow custom field editing control and validation. Using a key-in picture containing any of the alphabet indicators (@ ^ _) on a numeric entry field produces unpredictable results.

Using the Insert typing mode for a key-in picture could produce unpredictable results. Therefore, key-in pictures always receive data entry in Overwrite mode, even if the INS attribute is present.

Example:

Picture	Value Entered	Result String
@K###-##-####K	215846377	215-84-6377
@K#####|-####K	33064	33064
@K#####|-####K	330643597	33064-3597
@K<# ^^^ ##K	10AUG59	10 AUG 59
@K(###)@@@-##\@##K	305abc4555	(305)abc-45@55
@K###/?##-####K	7854555	000/785-4555
@k<#:##^Mk	530P	5:30PM
@K<#' <#"K	506	5' 6"
@K4#_#A-#K	1g12	41g1A-2

String Pictures

@Slength

@S All string pictures begin with @S.

length Determines the number of characters in the picture format.

A string picture describes an unformatted string of a specific length.

Example:

```
Name  STRING(@S20)   !A 20 character string field
```

APPENDIX B - CLARION FUNCTIONS

Mathematical Functions

ABS (return absolute value)

ABS(*expression***)**

ABS Returns absolute value.

expression A constant, variable, or expression.

The **ABS** function returns the absolute value of an expression. The absolute value of a number is always positive (or zero).

Return Data Type: REAL or DECIMAL

Example:

```
ABS(A - B)          ! Absolute value of the difference
```

INRANGE (check number within range)

INRANGE(*expression***,***low***,***high***)**

INRANGE Return number in valid range.

expression A numeric constant, variable, or expression.

low A numeric constant, variable, or expression of the lower boundary of the range.

high A numeric constant, variable, or expression of the upper boundary of the range.

The **INRANGE** function compares a numeric expression to an inclusive range of numbers. If the value of the expression is within the range, the function returns the value 1 for “true.” If the expression is greater than the high parameter, or less than the low parameter, the function returns a zero for “false.”

Return Data Type: LONG

Example:

```
If Formula:
INRANGE(Date % 7,1,5)        ! Is this a weekday?
```

INT (truncate fraction)

INT(*expression***)**

INT Return integer.

expression A numeric constant, variable, or expression.

The **INT** function returns the integer portion of a numeric expression. No rounding is performed, and the sign remains unchanged.

Return Data Type: REAL or DECIMAL

Example:

```
INT(8.5)     !  returns 8
INT(-5.9)    !  returns -5
```

LOGE (return natural logarithm)

LOGE(*expression***)**

LOGE Returns the natural logarithm.

expression A numeric constant, variable, or expression. If the value of the expression is less than zero, the return value is zero. The natural logarithm is undefined for values less than zero.

The **LOGE** (pronounced "log-e") function returns the natural logarithm of a numeric expression. The natural logarithm of a value is the power to which e must be raised to equal that value. The value of e is 2.71828182846.

Return Data Type: REAL

Example:

```
LOGE(2.71828182846)       !  returns 1
LOGE(1)                   !  returns 0
LOGE(Val)                 !  Get the natural log of Val
```

LOG10 (return base 10 logarithm)

LOG10(*expression***)**

LOG10 Returns base 10 logarithm.

expression A numeric constant, variable, or expression. If the value of the expression is zero or less, the return value will be zero. The base 10 logarithm is undefined for values less than or equal to zero.

The **LOG10** (pronounced “log ten”) function returns the base 10 logarithm of a numeric *expression*. The base 10 logarithm of a value is the power to which 10 must be raised to equal that value.

Return Data Type: REAL

Example:

```
LOG10(10)                !  returns 1
LOG10(1)                 !  returns 0
LOG10(Var)               !  The base ten logarithm of a variable named Var
```

RANDOM (return random number)

RANDOM(*low***,***high***)**

RANDOM Returns random integer.

Low A numeric constant, variable, or expression for the lower boundary of the range.

high A numeric constant, variable, or expression for the upper boundary of the range.

The **RANDOM** function returns a random integer between the *low* and *high* parameter values, inclusively. The low and high parameters may be any numeric expression, but only their integer portion is used for the inclusive range.

Return Data Type: LONG

Example:

```
RANDOM(1,49)               !Pick number between 1 and 49 for Lotto
```

ROUND (return rounded number)

ROUND(*expression***,***order***)**

ROUND Returns rounded value.

expression A numeric constant, variable, or expression.

order A numeric expression with a value equal to a power of ten, such as 1, 10, 100, 0.1, 0.001, etc. If the value is not an even power of ten, the next lowest power is used; 0.55 will use 0.1 and 155 will use 100.

The **ROUND** function returns the value of an *expression* rounded to a power of ten. If the *order* is a LONG or DECIMAL Base Type, then rounding is performed as a BCD operation. Note that if you want to round a real number larger than 1e30, you should use ROUND(num,1.0e0), and not ROUND(num,1). The ROUND function is very efficient ("cheap") as a BCD operation and should be used to compare REALs to DECIMALs at decimal width.

Return Data Type: DECIMAL or REAL

Example:

```
ROUND(5163,100)                 !returns 5200
ROUND(657.50,1)                 !returns 658
ROUND(51.63594,.01)             !returns 51.64
ROUND(Price / Rate,.01)         ! Round the commission to the nearest cent
```

SQRT (return square root)

SQRT(*expression***)**

SQRT Returns square root.

expression A numeric constant, variable, or expression. If the value of the expression is less than zero, the return value is zero.

The **SQRT** function returns the square root of the *expression*. If X represents any positive real number, the square root of X is a number that, when multiplied by itself, produces a product equal to X.

Return Data Type: REAL

Example:

```
SQRT(X^2 + Y^2)     ! The distance from 0,0 to x,y (pythagorean theorem)
```

Trigonometric Functions

Trigonometric functions return values representing angles and ratios of the sides of a right triangle. Angles are expressed in radians. PI is a constant which represents the ratio of the circumference and radius of a circle. There are 2*PI radians (or 360 degrees) in a circle.

The following equates provide high precision constants for PI and the conversion factors between degrees and radians. To use these conversion constants in ReportWriter, you can create computed fields and assign them these values.

```
PI       3.1415926535898    !The value of PI
Rad2Deg  57.295779513082    !Number of degrees in a radian
Deg2Rad  .0174532925199     !Number of radians in a degree
```

SIN (return sine)

SIN(radians)

SIN Returns sine.

radians A numeric constant, variable or expression for the angle expressed in radians.

The **SIN** function returns the trigonometric sine of an angle measured in *radians*. The sine is the ratio of the length of the angle's opposite side divided by the length of the hypotenuse.

Return Data Type: REAL

Example:

```
SIN(RadianField)                ! The sine of RadianField
```

COS (return cosine)

COS(*radians*)

COS Returns cosine.

Radians A numeric constant, variable or expression for the angle in radians.

The **COS** function returns the trigonometric cosine of an angle measured in *radians*. The cosine is the ratio of the length of the angle's adjacent side divided by the length of the hypotenuse.

Return Data Type: REAL

Example:

```
COS(RadianField)                ! The cosine of RadianField
```

TAN (return tangent)

TAN(*radians*)

TAN Returns tangent.

radians A numeric constant, variable or expression for the angle in radians.

The **TAN** function returns the trigonometric tangent of an angle measured in *radians*. The tangent is the ratio of the angle's opposite side divided by its adjacent side.

Return Data Type: REAL

Example:

```
TAN(RadianField)                ! The tangent of RadianField
```

ASIN (return arcsine)

ASIN(*expression*)

ASIN Returns inverse sine.

expression A numeric constant, variable, or expression for the value of the sine.

The **ASIN** function returns the inverse sine. The inverse of a sine is the angle that produces the sine. The return value is the angle in radians.

Return Data Type: REAL

Example:

```
ASIN(SineAngle)             ! The Arcsine of SineAngle
```

See Also: SIN

ACOS (return arccosine)

ACOS(*expression*)

ACOS Returns inverse cosine.

expression A numeric constant, variable, or expression for the value of the cosine.

The **ACOS** function returns the inverse cosine. The inverse of a cosine is the angle that produces the cosine. The return value is the angle in radians.

Return Data Type: REAL

Example:

```
ACOS(CosineAngle)           ! The Arccosine of CosineAngle
```

See Also: COS

ATAN (return arctangent)

ATAN(*expression*)

ATAN Returns inverse tangent.

expression A numeric constant, variable, or expression for the value of the tangent.

The **ATAN** function returns the inverse tangent. The inverse of a tangent is the angle that produces the tangent. The return value is the angle in radians.

Return Data Type REAL

Example:

```
ATAN(TangentAngle)          ! The Arctangent of TangentAngle
```

See Also: TAN

String Functions

ALL (return repeated characters)

ALL(*string* **[,***length***])**

ALL Returns repeated characters.

string A string expression containing the character sequence to be repeated.

length The length of the return string. If omitted the length of the return string is 255 characters.

The **ALL** function returns a string containing repetitions of the character sequence string.

Return Data Type: STRING

Example:

```
ALL('*',25)                    ! 25 asterisks
```

BEGINSWITH (evaluate beginning of string)

BEGINSWITH(*string*,*substring*)

BEGINSWITH Returns TRUE if the string begins with the substring.

string A string expression containing the character sequence to evaluate.

substring A string expression containing the character sequence to compare.

The **BEGINSWITH** function compares the beginning of the *string* with the *substring*. If the *string* begins with an exact match of the *substring*, the result will be TRUE. The BEGINSWITH function is case-sensitive. The function returns a BYTE containing zero (FALSE) or one (TRUE). Leading spaces are not ignored.

Return Data Type: BYTE

Example:

```
If Formula:
BEGINSWITH(CUS:Name,'Ajax')         ! Does Customer name start with 'Ajax' ?
```

CENTER (return centered string)

CENTER(*string* [,*length*])

CENTER Returns centered string.

string A string constant, variable or expression.

length The length of the return string. If omitted, the length of the string parameter is used.

The **CENTER** function first removes leading and trailing spaces from a *string*, then pads it with leading and trailing spaces to center it within the *length*, and returns a centered string.

Return Data Type: STRING

Example:

```
CENTER('ABC',5)          !returns ' ABC '
CENTER('ABC  ')          !returns ' ABC '
CENTER('     ABC')       !returns ' ABC '
```

CHR (return character from ASCII)

CHR(*code*)

CHR Returns the display character.

code A numeric expression containing a numeric ASCII character code.

The **CHR** function returns the character represented by the ASCII character *code* parameter.

Return Data Type: STRING

Example:

```
CHR(122)                 ! Get lower case z
CHR(65)                  ! Get upper case A
```

CLIP (return string without trailing spaces)

CLIP(*string*)

CLIP Removes trailing spaces.

string A string expression.

The **CLIP** function removes trailing spaces from a *string*. The return string is a substring with no trailing spaces. CLIP is frequently used with the concatenation operator in string expressions.

Return Data Type: STRING

Example:

```
CLIP(Last) & ', ' & CLIP(First) & Init & '.' !Full name in military order
```

CONTAINS(evaluate string)

CONTAINS(*string*,*substring*)

CONTAINS Returns TRUE if the string contains the substring.

string A string expression containing the character sequence to evaluate.

substring A string expression containing the character sequence to compare.

The **CONTAINS** function checks for the existence of the substring within the string. If the string contains an exact match of the substring, the result will be TRUE. The CONTAINS function is case-sensitive. The function returns a BYTE containing zero (FALSE) or one (TRUE).

Return Data Type: BYTE

Example:

```
IF Condition:
CONTAINS(CUS:Name,'oon')          ! Does Customer name contain with 'oon' ?
```

DEFORMAT (remove formatting from numeric string)

DEFORMAT(*string* [,*picture*])

DEFORMAT Removes formatting characters from a numeric string.

string A string expression containing a numeric string.

picture A picture token or the label of a CSTRING, STRING, or PSTRING variable containing a picture token. CSTRING is more efficient than a STRING or a PSTRING). If omitted, the picture for the string parameter is used. If the string parameter was not declared with a picture token, the return value will contain only characters that are valid for a numeric constant.

The **DEFORMAT** function removes formatting characters from a numeric string, returning only the numbers contained in the string. When used with a date or time *picture* (except those containing alphabetic characters), it returns a STRING containing the Clarion Standard Date or Time.

Return Data Type: STRING

Example:

```
'ATDT1' & DEFORMAT(Phone,@P(###)###-####P) & '<13,10>'
                                    ! Phone number for modem to dial
DEFORMAT(dBaseDate,@D1)             !Clarion Standard date from mm/dd/yy string
```

See Also: FORMAT

ENDSWITH (evaluate end of string)

ENDSWITH(*string*,*substring*)

ENDSWITH Returns TRUE if the string ends with the substring.

string A string expression containing the character sequence to evaluate.

substring A string expression containing the character sequence to compare.

The **ENDSWITH** function compares the end of the *string* with the *substring*. If the *string* ends with an exact match of the *substring*, the result will be TRUE. The ENDSWITH function is case-sensitive and ignores trailing spaces. The function returns a BYTE containing zero (FALSE) or one (TRUE).

Return Data Type: BYTE

Example:

```
If Formula:
ENDSWITH(CUS:Name,'ing')          ! Does Customer name start with 'ing' ?
```

FORMAT (format numbers into a picture)

FORMAT(*value*,*picture*)

FORMAT Returns a formatted numeric string.

value A numeric expression for the value to be formatted.

picture A picture token or the label of a STRING, CSTRING, or PSTRING variable containing a picture token (CSTRING is more efficient than STRING or PSTRING).

The **FORMAT** function returns a numeric string formatted according to the picture parameter.

Return Data Type: STRING

Example:

```
FORMAT(Emp:SSN,@P###-##-####P)  ! Format the soc-sec-nbr

FORMAT(DEFORMAT(Phone,@P###-###-####P),@P(###)###-####P)
                ! Change phone format from dashes to parens
```

See Also: DEFORMAT

INSTRING (search for substring)

INSTRING(*substring*,*string* [,*step*] [,*start*])

INSTRING Searches for a substring in a string.

substring A string constant, variable, or expression that contains the string for which to search.

string A string constant, or the label of the STRING, CSTRING, or PSTRING variable to be searched.

step A numeric constant, variable, or expression which specifies the step length of the search. A step of 1 searches for the substring beginning at every character in the string, a step of 2 starts at every other character, and so on. If step is omitted, the step length defaults to the length of the substring.

start A numeric constant, variable, or expression which specifies where to begin the search of the string. If omitted, the search starts at the first character position.

The **INSTRING** function steps through a *string*, searching for the occurrence of a *substring*. If the *substring* is found, the function returns the *step* number on which the *substring* was found. If the *substring* is not found in the *string*, INSTRING returns zero.

Return Data Type: LONG

Example:

```
INSTRING('DEF','ABCDEFGHIJ',1,1) ! returns 4
INSTRING('DEF','ABCDEFGHIJ',2,1) ! returns 0
INSTRING('DEF','ABCDEFGHIJ',2,2) ! returns 2
INSTRING('DEF','ABCDEFGHIJ',3,1) ! returns 2
```

See Also: BEGINSWITH, ENDSWITH, CONTAINS

LEFT (return left justified string)

LEFT(*string* [,*length*])

LEFT Left justifies a string.

string A string constant, variable, or expression.

length A numeric constant, variable, or expression for the length of the return string. If omitted, length defaults to the length of the string.

The **LEFT** function returns a left justified string. Leading spaces are removed from the string.

Return Data Type: STRING

Example:

```
LEFT(CompanyName)        !Left justify the company name
```

LEN (return length of string)

LEN(*string*)

LEN Returns length of a string.

string A string constant, variable, or expression.

The **LEN** function returns the length of a *string*. If the *string* parameter is the label of a variable, the function will return the declared length of the variable. Numeric variables are automatically converted to STRING intermediate values.

Return Data Type: LONG

Example:

```
If Formula:
  LEN(CLIP(Title) & ' ' & CLIP(First) & ' ' & CLIP(Last)) > 30
! is length more than 30 characters?

True Formula:
  CLIP(Title) & ' ' & SUB(First,1,1) & '. ' & Last    ! use first initial

Else Formula:
  CLIP(Title) & ' ' & CLIP(First) & ' ' & CLIP(Last)  ! use full name
```

LOWER (return lower case)

LOWER(*string*)

LOWER Converts a string to all lower case.

string A string constant, variable, or expression for the string to be converted.

The **LOWER** function returns a string with all letters converted to lower case.

Return Data Type: STRING

Example:

```
SUB(Name,1,1) & LOWER(SUB(Name,2,19))  !Make the rest of the name lower case
```

NUMERIC (check numeric string)

NUMERIC(*string*)

NUMERIC Validates all numeric string.

string A string constant, variable, or expression.

The **NUMERIC** function returns the value 1 (true) if the *string* contains a valid numeric value. It returns zero (false) if the *string* contains non-numeric characters. Valid numeric characters are the digits 0 through 9, a leading minus sign, and a decimal point.

Return Data Type: LONG

Example:

```
If Formula:
NUMERIC(PartNumber)                    ! If part number is numeric
```

RIGHT (return right justified string)

RIGHT(*string* [,*length*])

RIGHT Right justifies a string.

string A string constant, variable, or expression.

length A numeric constant, variable, or expression for the length of the return string. If omitted, the *length* is set to the length of the *string*.

The **RIGHT** function returns a right justified string. Trailing spaces are removed, then the string is right justified and returned with leading spaces.

Return Data Type: STRING

Example:

```
RIGHT(Message)                          !Right justify the message
```

SUB (return substring of string)

SUB(*string*,*position*,*length*)

SUB Returns a portion of a string.

string A string constant, variable or expression.

Position An integer constant, variable, or expression. If positive, it points to a character position relative to the beginning of the string. If negative, it points to the character position relative to the end of the string (i.e., a position value of -3 points to a position 3 characters from the end of the string).

length A numeric constant, variable, or expression of number of characters to return.

The **SUB** function parses out a sub-string from a *string* by returning *length* characters from the *string*, starting at *position*.

The SUB function is similar to the “string slicing” operation on STRING, CSTRING, and PSTRING variables, but is less flexible and efficient. “String slicing” is more flexible because it may be used on both the destination and source sides of an assignment statement, while the SUB function can only be used as the source. It is more efficient because it takes less memory than individual character assignments or the SUB function.

To take a “slice” of a string, the beginning and ending character numbers are separated by a colon (:) and placed in the implicit array dimension position within the square brackets ([]) of the string. The position numbers may be integer constants, variables, or expressions. If variables are used, there must be at least one blank space between the variable name and the colon separating the beginning and ending number (to prevent PREfix confusion).

Return Data Type: STRING

Example:

```
SUB('ABCDEFGHI',1,1)       ! returns 'A'
SUB('ABCDEFGHI',-1,1)      ! returns 'I'
SUB('ABCDEFGHI',4,3)       ! returns 'DEF'
```

UPPER (return upper case)

UPPER(*string*)

UPPER Returns all upper case string.

string A string constant, variable, or expression for the string to be converted.

The **UPPER** function returns a string with all letters converted to upper case.

Return Data Type: STRING

Example:

```
UPPER(Name)                !Make the name upper case
```

VAL (return ASCII value)

VAL(*character*)

VAL Returns ASCII code.

character A one-byte string containing a character.

The **VAL** function returns the ASCII code of a character.

Return Data Type: LONG

Example:

```
VAL('A')      ! returns 65
VAT('z')      ! returns 122
```

Bit Manipulation Functions

BAND (return bitwise AND)

BAND(*value*,*mask*)

BAND Performs bitwise AND operation.

value A numeric constant, variable, or expression for the bit *value* to be compared to the bit *mask*. The value is converted to a LONG data type prior to the operation, if necessary.

mask A numeric constant, variable, or expression for the bit *mask*. The *mask* is converted to a LONG data type prior to the operation, if necessary.

The **BAND** function compares the *value* to the *mask*, performing a Boolean AND operation on each bit. The return value is a LONG integer with a one (1) in the bit positions where the *value* and the *mask* both contain one (1), and zeroes in all other bit positions.

BAND is usually used to determine whether an individual bit, or multiple bits, are on (1) or off (0) within a variable.

Return Data Type: LONG

Example:

```
BAND(0110b,0010b)   ! returns 0010b     0110b = 6, 0010b = 2
```

BOR (return bitwise OR)

BOR(value,mask)

BOR Performs bitwise OR operation.

value A numeric constant, variable, or expression for the bit value to be compared to the bit mask. The value is converted to a LONG data type prior to the operation, if necessary.

mask A numeric constant, variable, or expression for the bit mask. The mask is converted to a LONG data type prior to the operation, if necessary.

The **BOR** function compares the value to the mask, performing a Boolean OR operation on each bit. The return value is a LONG integer with a one (1) in the bit positions where the value, or the mask, or both, contain a one (1), and zeroes in all other bit positions.

BOR is usually used to unconditionally turn on (set to one), an individual bit, or multiple bits, within a variable.

Return Data Type: LONG

Example:

```
BOR(0110b,0010b)    ! returns 0110b      0110b = 6, 0010b = 2
```

BXOR (return bitwise exclusive OR)

BXOR(*value*,*mask*)

BXOR Performs bitwise exclusive OR operation.

value A numeric constant, variable, or expression for the bit *value* to be compared to the bit *mask*. The *value* is converted to a LONG data type prior to the operation, if necessary.

mask A numeric constant, variable, or expression for the bit *mask*. The *mask* is converted to a LONG data type prior to the operation, if necessary.

The **BXOR** function compares the *value* to the *mask*, performing a Boolean XOR operation on each bit. The return value is a LONG integer with a one (1) in the bit positions where either the *value* or the *mask* contain a one (1), but not both. Zeroes are returned in all bit positions where the bits in the *value* and *mask* are alike.

BXOR is usually used to toggle on (1) or off (0) an individual bit, or multiple bits, within a variable.

Return Data Type: LONG

Example:

```
BXOR(0110b,0010b)   ! returns 0100b      0110b = 6, 0100b = 4, 0010b = 2
```

BSHIFT (return shifted bits)

BSHIFT(*value*,*count*)

BSHIFT	Performs the bit shift operation.
value	A numeric constant, variable, or expression. The *value* is converted to a LONG data type prior to the operation, if necessary.
count	A numeric constant, variable, or expression for the number of bit positions to be shifted. If *count* is positive, *value* is shifted left. If *count* is negative, *value* is shifted right.

The **BSHIFT** function shifts a bit *value* by a bit *count*. The bit value may be shifted left (toward the high order), or right (toward the low order). Zero bits are supplied to fill vacated bit positions when shifting.

Return Data Type: LONG

Example:

```
BSHIFT(0110b,1)     ! returns 1100b
BSHIFT(0110b,-1)    ! returns 0011b
```

Date / Time Procedures and Functions

Standard Date

A Clarion standard date is the number of days that have elapsed since December 28, 1800. The range of accessible dates is from January 1, 1801 (standard date 4) to December 31, 9999 (standard date 2,994,626). Date functions will not return correct values outside the limits of this range. The standard date calendar also adjusts for each leap year within the range of accessible dates. Dividing a standard date by modulo 7 gives you the day of the week: zero = Sunday, one = Monday, etc.

Standard Time

A Clarion standard time is the number of hundredths of a second that have elapsed since midnight, plus one (1). The valid range is from 1 (defined as midnight) to 8,640,000 (defined as 11:59:59:99). A standard time of one is exactly equal to midnight (which allows a zero value to be used to detect no time entered). Although time is expressed to the nearest hundredth of a second, the system clock is only updated 18.2 times a second (approximately every 5.5 hundredths of a second).

TODAY (return system date)

TODAY( )

The **TODAY** function returns the DOS system date as a standard date. The range of possible dates is from January 1, 1801 (standard date 4) to December 31, 2099 (standard date 109,211).

Return Data Type: LONG

Example:

```
TODAY() + 1  !Today's date plus one (tomorrow)
```

CLOCK (return system time)

CLOCK( )

The **CLOCK** function returns the time of day from the operating system time in standard time (expressed as hundredths of a second since midnight). Although the time is expressed to the nearest hundredth of a second, the system clock is only updated 18.2 times a second (approximately every 5.5 hundredths of a second).

Return Data Type: LONG

Example:

```
CLOCK()       ! The system time
```

DATE (return standard date)

DATE(*month*,*day*,*year*)

DATE Return standard date.

month A numeric constant, variable, or expression for the *month*.

day A numeric constant, variable, or expression for the *day* of the month.

year A numeric constant, variable or expression for the *year*. The valid range for a year value is 00 through 99 (which assumes the range 1900 - 1999), or 1801 through 2099.

The **DATE** function returns a standard date for a given *month*, *day*, and *year*. The *month* and *day* parameters allow out-of-range values. A *month* value of 13 is interpreted as January of the next year. A *day* value of 32 in January is interpreted as the first of February. Consequently, DATE(12,32,87), DATE(13,1,87), and DATE(1,1,88) all produce the same result.

Return Data Type: LONG

Example:

```
DATE(Hir:Month,Hir:Day,Hir:Year) !Compute hire date
```

See Also: Standard Date

DAY (return day of month)

DAY(*date*)

DAY Returns day of month.

date A numeric constant, variable, expression, or the label of a STRING, CSTRING, or PSTRING variable declared with a date picture token. The *date* must be a standard date. A variable declared with a date picture token is automatically converted to a standard date intermediate value.

The **DAY** function computes the day of the month (1 to 31) for a given standard date.

Return Data Type: LONG

Example:

```
DAY(TODAY())                !Get the day from today's date
DAY(TODAY()+2)              !Calculate the return day
```

See Also: Standard Date

MONTH (return month of date)

MONTH(*date***)**

MONTH Returns month in year.

date A numeric constant, variable, expression, or the label of a STRING, CSTRING, or PSTRING variable declared with a date picture token. The *date* must be a standard date. A variable declared with a date picture token is automatically converted to a standard date intermediate value.

The **MONTH** function returns the month of the year (1 to 12) for a given standard date.

Return Data Type: LONG

Example:

```
MONTH(DueDate)              ! The month from the date
```

See Also: Standard Date

YEAR (return year of date)

YEAR(*date*)

YEAR Returns the year.

date A numeric constant, variable, expression, or the label of a string variable declared with a date picture, containing a standard date. A variable declared with a date picture is automatically converted to a standard date intermediate value.

The **YEAR** function returns a four digit number for the year of a standard *date* (1801 to 2099).

Return Data Type: LONG

Example:

```
YEAR(LastOrd) < YEAR(TODAY())     ! Last order date not from this year
```

See Also: Standard Date

AGE (return age from base date)

AGE(*birthdate* [,*base date*])

AGE Returns elapsed time.

birthdate A numeric expression for a standard date.

base date A numeric expression for a standard date. If this parameter is omitted, the system date from DOS is used for the computation.

The **AGE** function returns a string containing the time elapsed between two dates. The age return string is in the following format:

```
1 to 60 days              - 'nn DAYS'
61 days to 24 months      - 'nn MOS'
2 years to 999 years      - 'nnn YRS'
```

Return Data Type: STRING

Example:

```
Emp:Name & 'is ' & AGE(Emp:DOB,TODAY()) & ' old today.'
```

DOS Procedures and Functions

MEMORY (return available memory)

MEMORY([*n*])

MEMORY Returns the amount of unused memory available.

n A numeric constant, variable, or expression. The valid range is 0 to 4, inclusive. If omitted, the default value is 0.

The **MEMORY** function returns the amount of unused memory available, in bytes. An *n* value of 0 or 1 returns the amount of conventional memory available, 2 returns the amount of EMS memory available, 3 returns total unused virtual memory, and 4 returns the largest virtual block of memory available. An out-of-range *n* value causes a return value of zero.

Return Data Type: LONG

Example:

```
MEMORY()                ! Return amount of unused conventional memory
MEMORY(0)               ! Return amount of unused conventional memory
MEMORY(1)               ! Return amount of unused conventional memory
MEMORY(2)               ! Return amount of unused EMS
MEMORY(3)               ! Return total unused virtual memory
MEMORY(4)               ! Return largest virtual block
```

PATH (return current DOS directory)

PATH()

The **PATH** function returns a string containing the current drive and directory.

Return Data Type: STRING

Example:

```
If Formula:
  PATH() = 'C:\'                      !If in the root

True Formula
  'You are in the Root Directory'     !  display message

Else Formula
 'You are not in the Root Directory'  !  display message
```

File Functions

EMPTY(evaluate record value)

EMPTY(‘*filelabel*’)

EMPTY Returns TRUE if a record exists.

filelabel The label of the file to evaluate.

The **EMPTY** function checks for the existence of a record at the detail level. If the *filelabel* contains a record when the detail prints, the result will be TRUE. The function returns a BYTE containing zero (FALSE) or one (TRUE).

This function is useful to provide a condition for a second detail band which you want to print only when no records exist for that level. For example, you want to print a band with “No orders for this customer” when a customer has no orders.

Return Data Type: BYTE

Example:

```
IF Condition:
EMPTY(‘Orders’)          ! Are there more Order records at this point?
```

APPENDIX C - DATABASE DRIVERS

Clarion ReportWriter for Windows achieves database independence with its built-in driver technology, enabling you to access data from virtually any file system and print reports from that data. Many file drivers are available and more are being added. All of the file drivers read and write in the file system's native format without temporary files or import/export routines.

Often, your Report Library's purpose is accessing data in its original format. For those times, you just import the file definition from the data file using the appropriate file driver.

ReportWriter accesses data from these different systems in the same manner; simply choose the correct file driver from the drop-down list when importing the file definition, and don't worry about it.

If you are deploying Report Libraries to end users, you must include the appropriate file driver(s) for the data files the Report Library accesses. Consult the appropriate section of this appendix for the Database Driver's filename.

To use a database driver, you must deploy the File Driver DLL for the format of your data files (one or more from the list below):

ASCII	**C60ASCX.DLL**
BASIC	**C60BASX.DLL**
Btrieve	**C60BTRX.DLL**
Clarion	**C60CLAX.DLL**
Clipper	**C60CLPX.DLL**
dBase III	**C60DB3X.DLL**
dBase IV	**C60DB4X.DLL**
DOS	**C60DOSX.DLL**
FoxPro	**C60FOXX.DLL**
ODBC	**C60ODBX.DLL**
Scalable	**C60SCAX.DLL**
TopSpeed	**C60TPSX.DLL**

ASCII Files

The ASCII driver reads standard ASCII files without field delimiters. This is often used for mainframe data import/export with an ASCII flat-file. A carriage-return/line-feed delimits records. The ASCII driver does not support keys.

ASCII database driver C60ASCX.DLL

Due to its lack of relational features and security (anyone can view and change an ASCII file using Notepad), it's unlikely you'll see the ASCII driver used to store large data files. But it can help you create a report form a text file.

Supported Data Types

STRING

GROUP

File Specifications/Maximums

File Size:	**4,294,967,295 bytes**
Records per File:	**4,294,967,295 bytes**
Record Size:	**65,520 bytes**
Field Size:	**65,520 bytes**
Fields per Record:	**65,520 bytes**
Keys/Indexes per File:	**n/a**
Key Size:	**n/a**
Memo fields per File:	**n/a**
Memo Field Size:	**n/a**
Open Data Files:	**Operating system dependent**

BASIC Files

The BASIC file driver reads comma-delimited ASCII files. Quotes (" ") surround strings, commas delimit fields, and a carriage-return/line-feed delimits records. The original BASIC programming language defined this file format. The Basic driver does not support keys.

The Basic file format provides a common file format for sharing data with spreadsheet and mail-merge programs. A common file extension used for these files is *.CSV, which stands for "comma separated values."

BASIC database driver C60BASX.DLL

Supported Data Types

BYTE	**DECIMAL**
SHORT	**PDECIMAL**
USHORT	**STRING**
LONG	**CSTRING**
ULONG	**PSTRING**
SREAL	**DATE**
REAL	**TIME**
BFLOAT4	**GROUP**
BFLOAT8	

File Specifications/Maximums

File Size:	**4,294,967,295 bytes**
Records per File:	**4,294,967,295 bytes**
Record Size:	**65,520 bytes**
Field Size:	**65,520 bytes**
Fields per Record:	**65,520 bytes**
Keys/Indexes per File:	**n/a**
Key Size:	**n/a**
Memo fields per File:	**0**
Memo Field Size:	**n/a**
Open Data Files:	**Operating system dependent**

Btrieve Files

This file driver reads Btrieve files using low-level direct access.

Under Clarion, the Btrieve file driver is implemented by using .DLLs and an .EXE supplied by Pervasive Software (formerly Btrieve Technologies, Inc.). For an application to use a Btrieve file driver, the following files must accompany the executable:

32-bit

You must purchase a 32-bit Btrieve engine from Pervasive Software.

A single file normally holds the data and all keys. Data filenames default to a *.DAT file extension. By default, the driver stores memos in a separate file, or optionally in the data file itself, given the appropriate driver string.

KEYs are dynamic, and automatically update when the data file changes.

INDEXes are stored separately from data files. INDEX files receive a temporary file name, and are deleted when the program terminates normally. INDEXes are static—they are not automatically updated when the data file changes. The application must create or update index files.

The Btrieve file format stores minimal file structure information in the file. The driver validates your description against the information in the file. It is possible to successfully open a Btrieve file that has key definitions that do not exactly match your definition.

Btrieve database driver	C60BTRX.DLL

Data Types

Clarion data type	Btrieve data type
BYTE	STRING (1 byte)
SHORT	INTEGER (2 bytes)
LONG	INTEGER (4 bytes)
SREAL	FLOAT (4 bytes)
REAL	FLOAT (8 bytes)
BFLOAT4	BFLOAT (4 bytes)
BFLOAT8	BFLOAT (8 bytes)
PDECIMAL	DECIMAL
STRING	STRING
CSTRING	ZSTRING
PSTRING	LSTRING
DATE	DATE
TIME	TIME
USHORT	UNSIGNED BINARY (2 bytes)
ULONG	UNSIGNED BINARY (4 bytes)
MEMO	STRING,LVAR or NOTE (see below)
BYTE,NAME('LOGICAL')	LOGICAL*
USHORT,NAME('LOGICAL')	LOGICAL*
PDECIMAL,NAME('MONEY')	MONEY*
STRING(@N0n-),NAME('STS')	SIGNED TRAILING SEPARATE*
DECIMAL*	

File Specifications/Maximums

File Size :	4,000,000,000 bytes
Records per File :	Limited by the size of the file
Record Size	
Client-based :	65,520 bytes variable length
Server based :	54K variable length
Field Size :	65,520 bytes
Fields per Record :	65,520 bytes
Keys/Indexes per File:	24 with NLM5 256 with NLM6. Client Btrieve v6.15

Page Size	Max Key Segments
512	8
1,024	23
1,536	24
2,048	54
4,096	119

This is the total number of components. If you have a multicomponent key built from three fields, this counts as three indexes when counting the number of allowed indexes.

Key Size :	255 bytes
Memo fields per File:	System memory dependent
Memo field size :	65,520 bytes
Open Files :	Operating system dependent

Clarion Files

The Clarion file driver is compatible with the file system used by Clarion for DOS 3.1 and Clarion Professional Developer 2.1.

Keys and Indexes exist as separate files from the data file. Keys are dynamic—they are automatically updated as the data file changes. The default file extension for a key file is *.K##. Indexes are static—they do not automatically update, the application must update them.

The driver stores records as fixed length. It stores memo fields in a separate file. The memo file defaults to the first eight characters of the File Label plus an extension of .MEM.

Clarion database driver C60CLAX.DLL

By avoiding the ASCII-only file formats of many other popular PC database application development systems, the Clarion file format provides a more secure means of storing data.

Data Types

BYTE	**DECIMAL**
SHORT	**STRING (255 byte maximum)**
LONG	**MEMO**
REAL	**GROUP**

If you do not have the file definition in a Clarion Data Dictionary or Report Library, use the File Import Utility in Clarion ReportWriter for Windows to define your files.

Maximum File Specifications

File Size:	**limited only by disk space**
Records per File :	**4,294,967,295**
Record Size:	**65,520 bytes**
Field Size :	**65,520 bytes**
Fields per Record:	**65,520 bytes**
Keys/Indexes per File:	**251**
Key Size :	**245 bytes**
Memo fields per File :	**1**
Memo Field Size:	**65,520 bytes**
Open Data Files:	**Operating system dependent**

Clipper Files

The Clipper file driver is compatible with Clipper Summer '87 and Clipper 5.0. The default data file extension is *.DBF.

Keys and Indexes exist as separate files from the data file. Keys are dynamic—they automatically update as the data file changes. Indexes are static—they do not automatically update, the application must update them. The default file extension for the index file is *.NTX.

The driver stores records as fixed length. It stores memo fields in a separate file. The memo file name takes the first eight characters of the File Label plus an extension of .DBT.

Clipper database driver C60CLPX.DLL

As a popular xBase database application development system, Clipper provides a common file format for many installed business applications and their data files. Use the Clipper driver to access these files in their native format.

Data Types

The xBase file format stores all data as ASCII strings. You may either specify STRING types with declared pictures for each field, or specify native Clarion data types, which the driver converts automatically.

Clipper data type	Clarion data type	STRING w/ picture
Date	DATE	STRING(@D12)
*Numeric	REAL	STRING(@N-_p.d)
*Logical	BYTE	STRING(1)
Character	STRING	STRING
*Memo	MEMO	MEMO

If your application reads and writes to existing files, a pictured STRING will suffice.

If you do not have the file definition in a Clarion Data Dictionary or Report Library, use the File Import Utility in Clarion ReportWriter for Windows to define your files.

File Specifications/Maximums

File Size:	**2,000,000,000 bytes**
Records per File:	**1,000,000,000**
Record Size:	**4,000 bytes (Clipper '87)**
	8,192 bytes (Clipper 5.0)
Field Size	
Character:	**254 bytes (Clipper '87)**
	2048 bytes (Clipper 5.0)
Date:	**8 bytes**
Logical:	**1 byte**
Numeric:	**20 bytes including decimal point**
Memo:	**65,520 bytes (see note)**
Fields per Record:	**1024**
Keys/Indexes per File:	**No Limit**
Key Sizes	
Character:	**100 bytes**
Numeric, Date:	**8 bytes**
Memo fields per File:	**Dependent on available memory**
Open Files:	**Operating system dependent**

Miscellaneous

Boolean Evaluation

Clipper allows a logical field to accept one of nine possible values (y,Y,n,N,t,T,f,F or a space character). The space character is neither true nor false. When using a logical field from a preexisting database in a logical expression, account for all these possibilities. Remember that when a STRING field is used as an expression, it is true if it contains any data and false if it is equal to zero or blank. Therefore, to evaluate a Logical field's truth, the expression should be true if the field contains any of the "true" characters (T,t,Y, or y). For example, if a Logical field were used to specify a product as taxable or nontaxable, the expression to evaluate its truth would be:

(If Condition):

```
Taxable='T' OR Taxable='t' OR Taxable='Y' OR Taxable='y'
```

Large MEMOs

Clarion supports MEMO fields up to a maximum of 64K. If you have an existing file which includes a memo greater than 64K, you can use the file but not the large MEMOs.

International Sort Sequence

Clarion' Clipper driver supports international sort orders, however, to maintain compatibility with Clipper's international sort order, remove the CLADIGRAPH= line from ..\CLARION6\BIN\C60ee.ENV file.

dBase III Files

The dBase3 file driver is compatible with dBase III. The default data file extension is *.DBF.

Keys and Indexes exist as separate files from the data file. Keys are dynamic—they automatically update as the data file changes. Indexes are static—they do not automatically update, the application must update them. The default file extension for the index file is *.NDX. International sort orders are supported.

The driver stores records as fixed length. It stores memo fields in a separate file. The memo file name takes the first eight characters of the File Label plus an extension of .DBT.

dBase III database driver C60DB3X.DLL

dBase III is probably the most common file format for PC database applications. These days, even desktop publishing programs can import dBase III compatible .DBF files. If the main task of your application is to export data files for other applications about which you know nothing, you should consider this format.

Data Types

The xBase file format stores all data as ASCII strings. You may either specify STRING types with declared pictures for each field, or specify native Clarion types, which the driver converts automatically.

dBase data type	Clarion data type	STRING w/ picture
Date	DATE	STRING(@D12)
*Numeric	REAL	STRING(@N-_p.d)
*Logical	BYTE	STRING(1)
Character	STRING	STRING
*Memo	MEMO	MEMO

If your application reads and writes to existing files, a pictured STRING will suffice.

If you do not have the file definition in a Clarion Data Dictionary or Report Library, use the File Import Utility in Clarion ReportWriter for Windows to define your files.

File Specifications/Maximums

File Size:	**2,000,000,000 bytes**
Records per File:	**1,000,000,000**
Record Size:	**4,000 bytes**
Field Size	
Character:	**254 bytes**
Date:	**8 bytes**
Logical:	**1 byte**
Numeric:	**20 bytes including decimal point**
Memo:	**64K (see note)**
Fields per Record:	**128**
Keys/Indexes per File:	**No Limit**
Key Sizes	
Character:	**100 bytes**
Numeric, Date:	**8 bytes**
Memo fields per File:	**Dependent on available memory**
Open Files:	**Operating system dependent**

Miscellaneous

Boolean Evaluation

dBase III allows a logical field to accept one of nine possible values (y,Y,n,N,t,T,f,F or a space character). The space character is neither true nor false. When using a logical field from a preexisting database in a logical expression, account for all these possibilities. Remember that when a STRING field is used as an expression, it is true if it contains any data and false if it is equal to zero or blank. Therefore, to evaluate a Logical field's truth, the expression should be true if the field contains any of the "true" characters (T,t,Y, or y). For example, if a Logical field were used to specify a product as taxable or nontaxable, the expression to evaluate its truth would be:

(If Condition):

```
Taxable='T' OR Taxable='t' OR Taxable='Y' OR Taxable='y'
```

Large MEMOs

Clarion supports MEMO fields up to a maximum of 64K. If you have an existing file which includes a memo greater than 64K, you can use the file but not modify the large MEMOs.

You can determine when your application encounters a large MEMO by detecting when the memo pointer variable is non-blank, but the memo appears to be blank. Error 47 (Bad Record Declaration) is posted, and any modification to the MEMO field is ignored.

International Sort Sequence

Clarion' dBaseIII driver supports international sort orders, however, to maintain compatibility with dBaseIII's international sort order, remove the CLADIGRAPH= line from \CLARION6\BIN\C60ee.ENV file.

dBase IV Files

The dBase4 file driver is compatible with dBase IV. The default data file extension is *.DBF.

Keys and Indexes exist as separate files from the data file. Keys are dynamic—they automatically update as the data file changes. Indexes are static—they do not automatically update, the application must update them. The default file extension for the index file is *.NDX.

dBase IV supports multiple index files, whose extension is *.MDX. The miscellaneous section describes procedures for using .MDX files.

The driver stores records as fixed length. It stores memo fields in a separate file. The memo file name takes the first eight characters of the File Label plus an extension of .DBT.

dBase IV database driver C60DB4X.DLL

dBase IV was never as widely adopted as dBase III. Choose this driver only when you must share data with an end-user using dBase IV.

Data Types

The xBase file format stores all data as ASCII strings. You may either specify STRING types with declared pictures for each field, or specify native Clarion types, which the driver converts automatically.

dBase data type	Clarion data type	STRING w/ picture
Date	DATE	STRING(@D12)
*Numeric	REAL	STRING(@N-_p.d)
*Logical	BYTE	STRING(1)
Character	STRING	STRING
*Memo	MEMO	MEMO

If your application reads and writes to existing files, a pictured STRING will suffice.

If you do not have the file definition in a Clarion Data Dictionary or Report Library, use the File Import Utility in Clarion ReportWriter for Windows to define your files.

File Specifications/Maximums

File Size:	**2,000,000,000 bytes**
Records per File:	**1,000,000,000**
Record Size:	**4,000 bytes**
Field Size	
Character:	**254 bytes**
Date:	**8 bytes**
Logical:	**1 byte**
Numeric:	**20 bytes including decimal point**
Float:	**20 bytes including decimal point**
Memo:	**64K (see note)**
Fields per Record:	**512**
Keys/Indexes per File:	
.NDX:	**No Limit**
.MDX	**47 tags per .MDX files**
Key Sizes	
Character:	**100 bytes**
Numeric, Date:	**8 bytes**
Memo fields per File:	**Dependent on available memory**
Open Files:	**Operating system dependent**

Miscellaneous

International Sort Sequence

dBase IV sorts as if there were no diacritics in a field, thus A is sorted the same as Ä. If two words are identical except for diacritic characters, then the words are sorted as though the diacritic character was greater than the normal character. For example Äa < Ab < Äb whereas a CLADIGRAPH of ÄAE will sort as Ab < Äa < Äb. Solution- if the same file is used in Clarion and dBase IV, issue a BUILD statement rebuild the keys before updating the file (reading the file causes no problems).

Boolean Evaluation

dBase IV allows a logical field to accept one of 11 possible values (1,0,y,Y,n,N,t,T,f,F or a space character). The space character is neither true nor false. When using a logical field from a preexisting database in a logical expression, account for all these possibilities. Remember that when a STRING field is used as an expression, it is true if it contains any data and false if it is equal to zero or blank. Therefore, to evaluate a Logical field's truth, the expression should be true if the field contains any of the "true" characters (T,t,Y, or y). For example, if a Logical field were used to specify a product as taxable or nontaxable, the expression to evaluate its truth would be:

(If Condition):

```
Taxable='1' OR Taxable='T' OR Taxable='t' OR Taxable='Y' OR Taxable='y'
```

Large MEMOs

Clarion supports MEMO fields up to a maximum of 64K. If you have an existing file which includes a memo greater than 64K, you can use the file but not modify the large MEMOs.

You can determine when your application encounters a large MEMO by detecting when the memo pointer variable is non-blank, but the memo appears to be blank. Error 47 (Bad Record Declaration) is posted, and any modification to the MEMO field is ignored.

DOS Files

The DOS file driver reads and writes any binary, byte-addressable files. Neither fields nor records are delimited. When reading a record, the driver reads the number of bytes defined in the file's RECORD structure, unless a length parameter is specified.

This file driver performs forward sequential processing only. No key or transaction processing functions are supported.

Due to its limitations, the main function of this driver is as a disk editor for binary files.

DOS database driver C60DOSX.DLL

Data Types

BYTE	**DECIMAL**
SHORT	**PDECIMAL**
USHORT	**STRING**
LONG	**CSTRING**
ULONG	**PSTRING**
SREAL	**DATE**
REAL	**TIME**
BFLOAT4	**GROUP**
BFLOAT4	

File Specifications/Maximums

File Size	**:**	**4,294,967,295**
Records per File	**:**	**4,294,967,295**
Record Size	**:**	**64K**
Field Size	**:**	**64K**
Fields per Record	**:**	**64K**
Keys/Indexes per File	**:**	**n/a**
Key Size	**:**	**n/a**
Memo fields per File	**:**	**n/a**
Memo Field Size	**:**	**n/a**
Open Data Files	**:**	**Operating system dependent**

FoxPro and FoxBase Files

The FoxPro file driver is compatible with FoxPro and FoxBase. The default data file extension is *.DBF.

The default index file extension is *.IDX. The default Memo file extension is .FBT. FoxPro also supports multiple index files, whose default extension is *.CDX. The miscellaneous section describes the procedures for using the .CDX files.

FoxPro database driver C60FOXX.DLL

Data Types

The xBase file format stores all data as ASCII strings.

FoxPro data type	Clarion data type	STRING w/ picture
Date	DATE	STRING(@D12)
*Numeric	REAL	STRING(@N-_p.d)
*Logical	BYTE	STRING(1)
Character	STRING	STRING
*Memo	MEMO	MEMO

If your application reads and writes to existing files, a pictured STRING will suffice.

If you do not have the file definition in a Clarion Data Dictionary or Report Library, use the File Import Utility in Clarion ReportWriter for Windows to define your files.

File Specifications/Maximums

File Size:	2,000,000,000 bytes
Records per File:	1,000,000,000 bytes
Record Size:	4,000 bytes
Field Size	
Character:	254 bytes
Date:	8 bytes
Logical:	1 byte
Numeric:	20 bytes including decimal point
Float:	20 bytes including decimal point
Memo:	65,520 bytes (see note)
Fields per Record:	512
Keys/Indexes per File:	No Limit
Key Sizes	
Character:	100 bytes (.IDX)
	254 bytes (.CDX)
Numeric, Date:	8 bytes
Memo fields per File:	Dependent on available memory
Open Files:	Operating system dependent

Miscellaneous

Boolean Evaluation

FoxPro and FoxBase allow a logical field to accept one of 11 possible values (0,1,y,Y,n,N,t,T,f,F or a space character). The space character is neither true nor false. When using a logical field from a preexisting database in a logical expression, account for all these possibilities. Remember that when a STRING field is used as an expression, it is true if it contains any data and false if it is equal to zero or blank. Therefore, to evaluate a Logical field's truth, the expression should be true if the field contains any of the "true" characters (1,T,t,Y, or y). For example, if a Logical field were used to specify a product as taxable or nontaxable, the expression to evaluate its truth would be:

(If Condition):

```
Taxable='1' OR Taxable='T' OR Taxable='t' OR Taxable='Y' OR Taxable='y'
```

Large MEMOs

Clarion supports MEMO fields up to a maximum of 64K. If you have an existing file which includes a memo greater than 64K, you can use the file but not modify the large MEMOs.

You can determine when your application encounters a large MEMO by detecting when the memo pointer variable is non-blank, but the memo appears to be blank. Error 47 (Bad Record Declaration) is posted, and any modification to the MEMO field is ignored.

ODBC—Open Data Base Connectivity

ODBC (Open DataBase Connectivity) is a Windows “strategic interface” for accessing data from a variety of Database Management Systems (DBMS) and data file formats, across a variety of networks and platforms. ODBC offers the same end result as the file driver libraries which ship with Clarion ReportWriter for Windows—file driver independence—though in a somewhat different manner.

The ODBC standard was developed and is maintained by Microsoft, which publishes an ODBC Software Development Kit (SDK) for use with its Office products, as well as its Visual Basic and Visual C++ products. ODBC support is another way in which Clarion ReportWriter for Windows provides an extensible platform for you to create reports.

ODBC database driver C60ODBX.DLL

ODBC Pro’s and Con’s

Using ODBC offers the following advantages:

ODBC is an excellent choice in a Client-Server environment, especially if the Server is a native Structured Query Language (SQL) DBMS. It allows you to use Client-Server support in your reports, without having to do much more than choose a file driver. ODBC was specifically designed to create a non-vendor-specific method of connecting front end applications to back end services. With ODBC, the Server can handle much of the work, especially for SQL JOIN and PROJECT operations, thereby speeding up processing.

Existing ODBC drivers cover a great many types of databases. There are ODBC drivers available for databases for which Clarion may not have a native driver—for example, Microsoft Excel and Lotus Notes files.

ODBC is already widespread. Major application suites such as Microsoft Office install ODBC drivers for file formats such as dBase and Microsoft Access. Keep in mind that many ODBC back end drivers have been updated and you should obtain the latest releases.

ODBC is platform independent. One of Microsoft’s prime objectives in establishing ODBC was to support easier access to legacy systems, or corporate environments where data resides on diverse platforms or multiple DBMS’s. As long as an ODBC driver and back end are available, it doesn’t matter whether you use Microsoft’s NetBEUI, SPX/IPX, DECNet or others; your Report Library can connect to the DBMS and access the data.

Given that there are many drivers available, and that the standard was developed by the company that developed Windows, you might consider using ODBC as the driver of choice for all your Windows applications. Yet, when deciding whether to use an ODBC driver or a Clarion ReportWriter for Windows native database driver, you must also consider possible **disadvantages**:

ODBC adds a layer—the ODBC Driver Manager—between ReportWriter and the database. When accessing files on a local hard drive, this generally results in slower performance. The driver manager must translate ReportWriter's ODBC API call to an SQL statement before any data access occurs.

ODBC uses SQL to communicate with the back end database. Although this can be very efficient when communicating with Client/Server database engines, it is normally less efficient than direct record access when using a file system designed around single record access, such as xBase or Btrieve.

The information required by the ODBC database manager to connect to a data source varies from one ODBC driver to another. Unlike the selection of Clarion file drivers, where file operations are virtually transparent, you may need to do some work to gather the information required to use a particular ODBC driver. Many ODBC drivers come with a Help (.HLP) file which documents special settings (usually stored in ODBC.INI); but the burden is on you to solve any problems with third-party ODBC drivers.

ODBC is not included with Windows. When distributing your reports, you'll need to install the ODBC drivers and the ODBC driver manager into the end user's system. This requires the ODBC SDK from Microsoft. In some cases, the back end server may have already provided a distribution kit which installs the ODBC driver on the workstation.

The normal Microsoft setup program that installs the ODBC driver manager adds an applet to the end user's Control Panel window for managing ODBC. It's very easy for an end user to use this tool to change the settings in the ODBC.INI file. The end user can unwittingly remove or modify the settings for the back end ODBC driver which would make it impossible for ReportWriter to connect to the data file. Additionally, since most ODBC drivers store the data directory in ODBC.INI, it's very easy for the end user to change it, again introducing a possible problem for ReportWriter .

Given the pros and cons, we recommend using the native Clarion ReportWriter for Windows file drivers when both a native driver and an ODBC driver exist for the same file format.

How ODBC Works

When you use ODBC to access data, four components must cooperate to make it work:

ReportWriter calls the ODBC driver manager, and sends it the appropriate requests for data, with the ODBC API.

Clarion does this for you transparently, using either the C60ODB.DLL (16-bit) application extension. When distributing your reports, be sure to include this file.

The ODBC driver manager receives the API calls, checks ODBC.INI for information on the data source, then loads the ODBC “back-end” driver.

The actual “interface” to the driver manager is a file called ODBCADM.EXE, which the Microsoft setup program places in the \Windows\System directory. This is the ODBC Administrator, which then loads other libraries to do its work.

The ODBC “back-end” driver is another library (.DLL) which contains the executable code for accessing the data.

Various third-parties supply “back-end” drivers. For example, Lotus Development Corp. supplies the ODBC driver for Lotus Notes. Microsoft Office distributes an ODBC SDK containing drivers for most of their database products.

The data source is either a data file (usually when ODBC is used for local data access), or a remote DBMS, such an Oracle 7 database.

The data source has a descriptive name; for example, “Microsoft Access Databases.” The name serves as the section name in the ODBC.INI file.

The ODBC driver manager must know the exact data source name so that it can load the right driver to access the data. Therefore, it’s vitally important that you know the precise data source name.

Importing ODBC File Definitions—the Basics

Using ODBC data sources only requires choosing Clarion ReportWriter's ODBC driver and importing the file definition. You may also need to provide the parameters in the OWNER and NAME attributes of the FILE declaration.

When creating a report for ODBC tables, importing the file definitions provides this information in the appropriate fields.

The following introduces the basics. Of course, you must also be sure that the field data types in your dictionary match the variable formats supported by the DBMS you're connecting to.

1. Create a new Report Library file.
2. In the **Create Using** field, choose ***Database***.
3. Press the ellipsis (...) button to select the ODBC data source.

   The *Select File Driver* dialog appears.
4. Select **ODBC** from the drop-down list, then press the **OK** button.

   The Data Sources dialog appears. This is similar to the ODBC Administrator's interface. If the data source has not yet been defined, you can add it by pressing the **New** button.
5. Highlight a Data Source, then press the **Next** button.
6. If the Data Source has password protection, the Logon dialog appears. Provide the User ID and password, then press the **OK** button.

   If the file contains multiple tables, the *Tables for ...* dialog appears.
7. Highlight a table, then press the **Finish** button.

   The file definition is imported and is available for your report.
8. Repeat the last seven steps for each table in the database.

Data Types

ODBC data type	Clarion data type
CHAR	STRING, CSTRING
VARCHAR	STRING, CSTRING
LONG VARCHAR	STRING, CSTRING
DECIMAL	DECIMAL, BYTE[1],SHORT[1],LONG[1],PDECIMAL
NUMERIC	PDECIMAL, BYTE[1],SHORT[1],LONG[1],DECIMAL
SMALLINT	SHORT
INTEGER	LONG
REAL	SREAL
FLOAT	REAL
DOUBLE PRECISION	REAL
BIT	BYTE
TINYINT	BYTE
BIGINT DECIMAL,	PDECIMAL
BINARY	STRING
VARBINARY	STRING
LONGBINARY	STRING
DATE	DATE
TIME	TIME
TIMESTAMP	STRING[2]

Notes:

1 Clarion LONG, SHORT, and BYTE can be used with ODBC DECIMAL and NUMERIC data types if the ODBC field does not have any decimal places.

2 ODBC TIMESTAMP fields can be manipulated using a STRING(8) followed by a GROUP over it which contains only a DATE field and a TIME field.

Example:

```
TimeStampField      STRING(8),NAME('TimeStampField')
TimeStampGroup      GROUP,OVER(TimeStampField)
TimeStampDate             DATE
TimeStampTime             TIME
                    END
```

CREATE() will create a TIMESTAMP field if the you use a similar structure.

Note: Your back-end database may contain data types that are not listed here. These data types are converted to ODBC data types by the back-end database. Consult you back-end database's documentation to determine which ODBC data type is used.

Scalable SQL

Scalable SQL Server

For complete information on the Scalable SQL database system, please review Pervasive Software's documentation.

Scalable database driver: C60SCAX.DLL

Scalable SQL Driver

The Scalable SQL Driver is one of several SoftVelocity SQL Accelerator drivers. These SQL Drivers share a common code base and many common features such as SoftVelocity's unique, high speed buffering technology, common driver strings, and SQL logging capability.

The Scalable SQL Driver converts standard Clarion file I/O statements and function calls into optimized SQL statements, which it sends to the backend Scalable SQL server for processing.

SQL Import Wizard—Login Dialog

The Import Wizard lets you import Scalable SQL table definitions into your Report Libraries. When you select the Scalable SQL Accelerator Driver from the driver drop-down list, the Import Wizard opens the *Login/Connection* dialog. The *Login/Connection* dialog collects the connection information for the Scalable SQL database.

Fill in the following fields in the *Login/Connection* dialog:

Database Name

Select the Scalable SQL database that contains the tables to import. If the **Database Name** list is empty, you may type in the name.

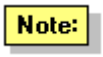
See your DBA or your server documentation for information on how the database is specified.

DDF Directory

Press the **Browse** button to select the pathname or directory containing the database DDF files.

Database Directory

Press the **Browse** button to select the pathname or directory containing the database.



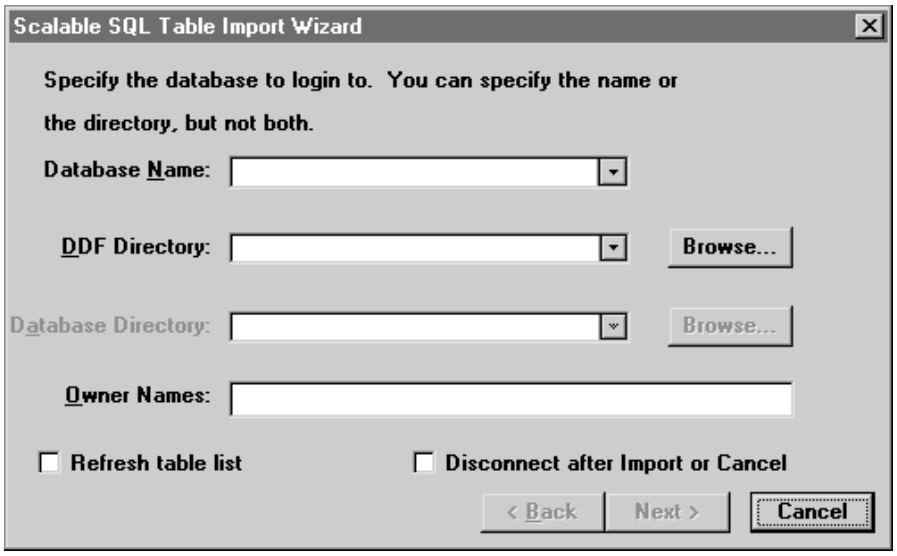
You may specify either the Database Name or the DDF directory, but not both.

User Name

The user login ID for the named database.

Password

The user password for the named database.



Owner Names

Optionally, type a comma separated list of names the Scalable SQL driver tries when opening encrypted Btrieve files. If a name contains a comma or space, it must be surrounded by single quotes.

Refresh table list

Check this box to refresh the list of tables to import when you press the Next > button. Clear the box to improve performance when the database is likely to be unchanged between imports.

Disconnect after Import or Cancel

Check this box to disconnect from the server after importing the (last) definition. Generally, you should clear this box when importing multiple definitions in order to maintain your connection to the server between imports.

Next >

Press this button to open the Import Wizard's Import List dialog.

SQL Import Wizard—Import List Dialog

When you press the Next > button, the Import Wizard opens the Import List dialog. The Import List dialog lists the importable items. Highlight the table whose definition to import, then press the Finish button to import. The Import Wizard adds the definition to your Clarion Data Dictionary, then opens the File Properties dialog to let you modify the default definition. Import additional tables by repeating these steps.

TopSpeed Files

The TopSpeed Database file system is a high-performance, high-security, proprietary file driver for Clarion development tools. It is not file compatible with the Clarion file driver's data.

Data tables, keys, indexes and memos can all be stored together in a single DOS file. The default file extension is *.TPS. A separate "Transaction Control File" takes the *.TCF extension.

The TopSpeed driver can optionally store multiple tables in a single DOS file. This allows you to open as many data tables, keys, and indexes as necessary using a single DOS file handle. This feature may be especially useful when there are a large number of small tables, or when a group of related files are normally accessed together. All keys, indexes, and memos are stored internally.

In addition, the TopSpeed file system supports the BLOB data type (Binary Large OBject), a field which is completely variable-length and may be greater than 64K in size. A BLOB must be declared before the RECORD structure. Memory for a BLOB is dynamically allocated and de-allocated as necessary. For more information, see BLOB in the Language Reference.

TopSpeed database driver C60TPSX.DLL

This new driver offers speed, security, and takes up fewer resources on the end user's system.

Data Types

BYTE	**DECIMAL**
SHORT	**STRING**
USHORT	**CSTRING**
LONG	**PSTRING**
ULONG	**MEMO**
SREAL	**GROUP**
REAL	**BLOB**

If you do not have the file definition in a Clarion Data Dictionary or Report Library, use the File Import Utility in Clarion ReportWriter for Windows to define your files.

Maximum File Specifications

```
File Size           :     Limited only by disk space
Records per File    :     Unsigned Long (4,294,967,295)
Record Size         :     15K
Field Size          :     15K
Fields per Record   :     15K
Keys/Indexes per File:    240
Key Size            :     64K
Memo fields per File:     255
Memo Field Size     :     64K
BLOB fields per File:     255
BLOB Size           :     Hardware dependent
Open Data Files     :     Operating system dependent
```

Importing File definitions from a Multiple Table .TPS File

TopSpeed Database files can store multiple tables in a single file.

The data files share a single DOS file handle, opened when the first file is opened, and closed when the last file is closed.

This feature is especially useful when there are a large number of small tables, or when the application must normally access group of related files together.

To import a TopSpeed File:

1. Choose File ▸ Import ▸ TopSpeed File.

   The *Open TopSpeed File* window appears.

2. Press the ellipsis (…) button next to the **Filename** entry box.

   A standard file open dialog appears.

3. Select the file from the standard file open dialog, then press the **OK** button (if necessary, “walk” the directory tree to the appropriate subdirectory).

   If the file contains multiple tables, the **Tables for ...** dialog appears.

4. Highlight a table, then press the **Select** button.

5. Press the **OK** button on the *Open TopSpeed File* window.

   The file definition is imported and is available for your report.

6. Repeat the these steps for each table in the database.

APPENDIX D - GLOSSARY

All definitions should be considered general terms, except where otherwise indicated. The context for definitions marked (Clarion) pertain to the Clarion language or the Clarion development environment. Likewise for (SQL), which applies to generalized Structured Query Language usage.

access key (Clarion) A specified key or index to set the order for processing records in a procedure.

Alias An alternate name for a data file, which allows multiple, independent operations on it. Clarion provides a separate record buffer for each alias, increasing the performance of the separate operations.

ANSI character set
Character set standardized by the American National Standards Institute. Many ANSI characters are different then the corresponding ASCII character set. The ANSI set contains more non-English characters. The standard Microsoft Windows character set is the ANSI character set.

append Add a record to a data file, usually without updating a key or index.

application A computer program designed for a specific type of work; the terms “application” and “program” are interchangeable. In general, when referring to a Windows program, “application” is the preferable term.

array A ordered series or group of dimensioned values or data items.

ASCII character set
Character set standardized as the American Standard Code for Information Interchange. The standard IBM PC character set.

assignment statement
A statement placing a value in a variable; for example, A = 6 places the value 6 in variable “A.”

auto-increment field
(Clarion) A key field which stores a value which increases with each successive record, and is generally not available to the end user. The application places the value in the field immediately upon appending the record.

band An element of the report which prints at a specific time.

binary memo (Clarion) A memo field suitable for holding non-ASCII contents, such as images.

bind (Clarion) A statement which allows a variable name to be used in a dynamic expression which is assembled and processed at run-time.

bitmap A binary file representation of a graphic or picture; raster format defines the image by absolute pixels. Popular bitmap formats supported by Clarion include .BMP, .GIF, .ICO, .PCX, .JPG. Sometimes refers specifically to the .BMP file format, an uncompressed, but widely supported file format.

BLOB (Binary Large OBject) A variable length field which may exceed the 64K limit and is suitable for holding non-ASCII contents, such as images.

Boolean A logical expression which evaluates to true or false, one or zero.

Border or Line Color
The color designated for the outside line of a graphical control.

break field (Clarion) A field or variable monitored when processing a report structure. When the value in the field changes while sequentially processing records, the print engine processes the next element in the report structure (usually the group footer).

case sensitive A characteristic indicating whether a command treats text typed with capital (uppercase) letters differently than those typed with lower case, or a combination of both.

case structure A control structure which branches execution to a statement (or group of statements) based upon a single condition or expression.

character string
An alphanumeric data type.

check box A control consisting of a small square or diamond, in which an end user indicates a on/off, yes/no, or true/false choice.

Clarion standard date
(Clarion) The number of days elapsed since December 28, 1800; the valid range is from Jan. 1, 1801 through Dec. 31, 2099.

click To place the mouse pointer on a control or window, then press and release the left mouse button.

client A system attached to a network that accesses shared network resources.

client application
A program that makes requests of a server application using a defined interface such as DDE, RPC, or NetBIOS.

client server architecture
A network configuration by which linked workstations request services from a dedicated program running on a server.

client server networking
A network architecture in which shared resources are concentrated on powerful server machines and the attached desktop systems fulfill the role of clients, making requests across the network for centralized information.

clipboard A temporary storage area in memory for holding data, maintained by Windows.

color dialog Standard Windows dialog for choosing color.

column (SQL) Generally refers to a list of database field contents arranged by records.

common file dialog
A standard Windows dialog for displaying drives, directories, and file names.

concatenate Append two string data elements to form a longer string comprised of both.

conditional statement
An IF statement which branches subsequent execution based on a logical condition.

constant A static value.

control alignment
The “Snap-To” behavior, as found in the Report Formatters, by which you may “line up” report elements.

control properties
(Clarion) Attributes which determine the appearance and functionality of a report control.

criteria (SQL) An expression containing a condition which limits the records for processing.

current directory
The default DOS subdirectory, in which Windows or DOS searches for files not identified with a fully qualified file name.

current record (Clarion) The current database record in the record buffer.

cursor The mouse pointer. Changing the cursor “shape” can indicate the type of action or selection the end user can effect on a given control or window.

data dictionary (Clarion) ASCII file describing the individual data files which comprise the database, their structure, keys, relations, and other information describing how an application will process the contents of the database.

data file Generally, a collection of data elements in an organized format, usually arranged by records (rows) and fields (columns).

data type A physical description of the type of storage supported by a variable; what sort of values it can hold.

database A structured collection of data, contained in one or more data files, plus the key files and other information which describes the order and relations of the data elements.

database administrator
(DBA) A person responsible for designing and maintaining a multi-user database system.

database definition file
(*.DDF). A Btrieve file, separate from the data file, containing the database structure. Equivalent to the header contained internally in most other PC database file formats.

database design
The process of planning and describing the most efficient application or system for storing and managing data for a specific project.

database driver
A collection of functions and procedures contained in a dynamic link library, supporting low level access to a specific database file format.

database integrity
Under the relational model, database integrity consists of two general rules:

1. Each database file or table must have a primary key serving as a unique identifier for all records.

2. When a table has a foreign key matching the primary key of another table, each value in the foreign key must either equal a value in the primary key of the other table, or be null.

dBase format PC database file format popularized by dBase III.

DBMS Database Management System: generic term for a program that enables a system to perform all the functions associated with managing a database.

default An assumed state or action, which the end user accepts or executes with little or no action.

delimiter A character marking the boundaries of one database field from another.

dependent entity
(SQL) A set of data elements dependent on other related entities in the database to identify them..

desktop The screen area in which all windows, dialog boxes, and icons appear.

DETAIL structure
(Clarion) The portion of a report structure which usually conveys the main data within the printed report. The application loops through, updates, and prints the detail band controls with the contents of all the records being processed.

disabled A window, menu, or control visible but prevented from gaining focus.

double-click To press and release the left mouse button twice, quickly. Executes the default action on a selection.

drag To press the left mouse button, then move the mouse while continuing to hold the button down. Usually a visual cue indicates a process such as moving a selected object, or rubber-banding a region. Releasing the button completes the action.

drag and drop To select an object in a window or dialog box, press down the left mouse button, move the mouse while continuing to hold the button down, then release the button when the pointer is on top of another object. When drag and drop is supported by the program(s), the action generally indicates the dropped object is to be processed in some way by the recipient object.

drop down list A list box control which only displays only the current selection when closed. When the user opens the list box, it expands to include additional choices.

dynamic link library (DLL) A library of shared functions that applications link to at run-time, as opposed to compile time.

encryption The storage on disk of data in scrambled or encrypted form, such that an unauthorized user may not access the data in an intelligible format.

Excel format File format used by the Microsoft Excel spreadsheet application. Note: an ODBC driver exists for this format, and is available in the Microsoft ODBC 2.0 Software Development Kit.

exclusive access Opening a DOS file so that no other user in a multi-user environment may update the same file.

executable A standard .EXE application file capable of being launched by the Microsoft Windows shell.

expression A mathematical formula containing any valid combination of variables, functions, operators, and constants.

extension A file name suffix; up to three characters in the DOS file system. Windows 3.1 matches document files to their application via the [Extensions] section in the WIN.INI file.

external name (Clarion) An attribute which holds the native format name (such as a DOS file name) for a given data element. ReportWriter refers to the file by the Clarion label.

field A basic data element or category which names all the values in a column of data within a database file or table.

file handle An operating system pointer to a file. The “FILES=“ line in the CONFIG.SYS file sets the system limit on the total number of allowable open files at one time.

fill color The color designated for the inside of a graphical control.

filter An expression which isolates a subset of records for an operation.

folder A logical container implemented by the shell, within which the user may group a collection of items. Analogous to a file directory.

font The family name of related type face files. For example, "Times New Roman" is the font name, and "Times New Roman plain," "Times New Roman Italic," "Times New Roman Bold," and "Times New Roman Bold Italic" are the styles, which are stored in separate files.

font dialog A standard Windows dialog for picking a typeface, style, size, and optionally, the text color.

font style Character formatting applied to a font face, such as bold, italic, or bold italic.

foreign key (SQL) A key in one table (database file) whose values match the primary key of another table.

form A window that displays a single record for editing. By convention there is a separate entry box for each field displayed, and fields are stacked in a vertical arrangement.

form letter A mail merge document containing "boiler-plate" text, in which controls reference fields from which to obtain information when creating letters to individuals.

form report style A report format generally containing field labels and values arranged in a vertical format.

formatter (Clarion) A specialized window which allows you to visually define the formatting for a report in "WYSIWYG" fashion.

function (Clarion) A specialized procedure which returns a value. The function declaration may optionally define parameters which are passed when calling the function. A function may be used within computed or conditional fields.

GIF image Graphics Interchange File format; an image format popularized by CompuServe. Generally acknowledged to offer the best compression ration for 256 color or less images. Attention: should you use the word "GIF" anywhere within an application or program, you must add a trademark notice: "GIF (Graphics Interchange Format) is a trademark of CompuServe Information Services."

grayed A visual cue to the user that the window, menu, or control is unavailable or disabled.

grid snap A series of coordinates, represented by dots, such as those used by the Report Formatter, to force controls to exact positioning.

groupbox A rectangular line frame with a label at upper left, used to define related controls.

I-beam A special cursor usually indicating the end user can type text into an edit control.

I/O Input/Output. The process of moving information into and out of the system.

icon A graphical representation of a physical object in the system, such as a printer. Also, any small image representing an action, concept or program, as when an icon appears on a command button. The normal icon file format carries the .ICO extension; one of its main features is built-in support for transparency.

identifier A label uniquely identifying a variable or other program element.

independent entity
(SQL) A set of data elements sharing a set of properties independent of other related entities in the database. Independent entities have unique identifiers, and therefore, primary keys.

index file An external key file ordered according to the contents of a specified field or expression. An index file usually must be manually updated when adding, deleting, or changing records.

INI file A Windows Initialization file in ASCII format. The .INI file is divided into sections separated by an identifier enclosed in square brackets. Variables and their values follow, each pair separated by a carriage return, with an equal sign between the variable name and its value. Values may be stored as strings or integers.

insertion point The point in a document at which the next characters typed by the end user will appear.

interface The communication between the computer and the user; it presents information to the user and accepts the user's input.

ISAM Indexed Sequential Access Method; a database organization in which data files are ordered by keys, and may be retrieved in the sequence of the keys.

join A join takes two database files (or tables) and creates a new, wider table consisting of all possible concatenated records (or rows).

JPG image A true-color graphics file format featuring 24-bit color storage. It usually provides for adjustable lossy compression, which allows for greater compression but loss of some resolution.

key An indexed file ordered according to the contents of a specified field or fields. Keys are usually dynamically updated whenever the value in a key field changes.

key-in template picture
(Clarion) A formatting option, which restricts and verifies end user keyboard input according to a specified character pattern applied upon a variable.

keyboard accelerator
A hot-key combination which directly executes a command.

label (Clarion) A unique identifier for a variable, procedure, function, routine, or data structure.

lock A concurrency control mechanism to prevent more than one user from updating the same record at the same time.

locked field or record
A field or record currently being updated by one user within a multi-user database, such that an attempt by another user to update the same record at the same time will fail.

logical operator
A true/false or bitwise comparison of two values; logical operators are: =, >, <, <>, >=, <+, NOT, AND, OR, and XOR.

lookup table or lookup file
A database file on one side of a one to many relation, upon which a variable is searched for, and a corresponding field in the related table is returned.

many-to-many relationship
A connection between two data entities in which there may exist many corresponding values in the foreign key in one database file or table, to many corresponding values in the foreign key of another table. Usually implemented via a "join" file breaking them into two 1:Many relations.

many-to-one relationship
A connection between two data entities in which there may exist many corresponding values in the foreign key in one database file or table, to only one value in the primary key of another "look-up" table. The relationship implicitly describes the direction of the relation. Also called a child-parent relation.

memo A free-form, variable length text field, suitable for storing very long strings. In most PC file formats, the memo is stored in a file separate from the fixed-length database fields. A binary memo field is a specialized type of memo field suitable for storing binary information such as graphics.

message box A standard windows element, usually consisting of a short message string, an OK button, often a standard icon such as "stop" or "information." It may optionally contain additional buttons such as "Cancel," and "Retry."

metafile In Windows, the representation of a graphic or line art in vector (device independent) format; defines the image as a series of lines and curves, allowing for smooth resizing. Clarion supports the .WMF (Windows Metafile) vector format. The metafile is actually a stored collection of the commands which instruct the GDI (Windows Graphics Device Interface) to display the graphic on the output device.

minimize box
A window control which resizes a window to iconic size, usually at the bottom of the desktop, or if a child window, to iconic size, usually at the bottom of the application window.

mnemonic access key
The underlined letter in the command names on Microsoft Windows menus. When a user activates a pull down menu, the key executes the command.

multi-user database
A database system designed so that more than one user can access a file or record at the same time. The system requires concurrency checking so that two users don't attempt to update the same record at the same time.

natural join (SQL) A join which takes two database files (or tables) and creates a new, wider table consisting of all possible concatenated records (or rows), where the new table contains two identical columns, one of which is dropped.

nested queries (SQL) A single query consisting of both an outer and inner query. Allows for more efficient retrieval of data from large tables by combining multiple operations into one.

nesting Placing one operation inside another, such as nesting a function within another by specifying the nested function as a parameter of the first.

normalization The representation of data entities in their simplest forms, for the purpose of quickest access and most efficient storage. The normalization process includes the elimination of redundant data groups, and the elimination of redundant data elements.

null value A zero or empty value.

ODBC The Open Database Connectivity standard supported by many Windows applications. Provides a standard API for accessing multiple database file formats via replaceable file drivers, and Client/Server support. The ODBC SDK is published by Microsoft.

ODBC Administrator
A redistributable Microsoft application for adding, maintaining or deleting individual ODBC drivers within a system. Usually located in the Windows\System directory, the executable file name is ODBCADM..EXE.

ODBC Control Panel applet
A Windows Control Panel interface to the ODBC administrator.

ODBC driver A driver library containing the individual functions supporting standard ODBC calls for a particular file format.

one-to-many relationship
A connection between two data entities in which there may exist one corresponding value in the primary key of one database file or table, to many identical values in the foreign key of another table. The relationship implicitly describes the direction of the relation. For example, the relation of states to cities implies a state may have many cities. Also called a parent-child relation.

one-to-one relationship
A connection between two data entities in which there may exist one and only one corresponding value in the primary key of one database file or table, to a single identical value in the foreign key of another table. For example, the relation of customer name to internet address. The data is usually split into two separate tables for storage savings; all customers have names, but only a minority have internet addresses.

origin The upper left corner of a window or control, expressed in x,y coordinates (0,0).

orphan A portion of text or data separated from its complementary preceding data by a page break.

outer join (SQL) A join which includes all records from one database file, and only those records from another in which the values in a selected field (or fields) match those in the first.

overlay (Clarion) A variable or field sharing the same location as another. Acts as a data "re-declaration, and provides more efficient storage. Most useful in "either/or" situations when a variable and its overlay are of similar types but use different pictures.

page footer The section of a report composed after the last detail that will fit on a page has been composed.

Page Form (Clarion) A report element defined once, when first composing the report, then printed on all pages of the report.

page header The section of a report composed before the first detail to print on a page.

page overflow In Clarion, the point at which the report library composes enough data to complete a page; the library will either send the page to the Windows spooler at that point, or first check to verify there are no "widows," if the application so specifies.

palette The table of available colors which a given window may user for painting.

parameter An argument or optional variable passed to a procedure.

PCX image A standard graphics file format, offering moderate compression, originally developed by the Zsoft corporation. The Windows Paintbrush accessory supports this format.

pel Equivalent to pixel; abbreviation for picture element. The smallest screen unit addressed in graphic mode; a dot.

picture token (Clarion) A formatting string, which specifies a specific "picture" or masking format for displaying and editing variables. The picture token begins with the "@" character.

pixel Equivalent to pel; abbreviation for picture element. The smallest screen unit addressed in graphic mode; a dot.

point size A measurement expressed in points; one point equals 1/72nd inch, or 1/28 centimeter.

pointer The mouse cursor. Or, an index entry which locates or "points" to the corresponding data record.

popup menu A menu that appears disconnected from other visual elements. Windows 95 and Clarion frequently displays popup menus when the user clicks the right mouse button. By convention, the menu is associated with the item clicked on.

prefix (Clarion) A short identifying string for a data structure. Provides a method for resolving variable names when, for example, two database files include fields whose names are the same.

primary key (SQL) A database field or expression which uniquely identifies each record in the table or database file.

print job One complete task sent to the Windows print spooler (accessible from Print Manager).

print structure (Clarion) The parts of a report structure, which include the group break structure, detail, header, footer, and form.

printer driver An external library file containing low level instructions and functions by which the Windows GDI library sends specific commands to the printer.

printer font A typeface resident in the printer's RAM.

progress bar A control that displays a graphic representation of a dynamic value by progressively coloring in a rectangle as the value changes.

prompt A text label which normally appears near a screen control, to identify the control.

property An attribute of a window, control, or other object.

property list A dialog intended to allow the convenient grouping of closely related items in a single place.

query (SQL) An operation upon a database table which results in another table or subset of the first.

range constraint
A bounds for a database operation limiting the operation to a set of records for which a given field falls within specified starting and ending values.

RECORD (Clarion) A data structure representing one row in a database table.

referential integrity
The process by which an application "follows through" on an update to a key field in one file, to check its related record in another file. This maintains valid parent-child relationships within the database.

relationship A logical link between records in data files based upon a duplicate (linking) field.

restore button A window control which resizes a window from a maximized state to the last size prior to maximizing.

rich text format (RTF)
A common word processing file format, originally designed for transportability between word processing systems across different operating systems. The default format for the source document for the Windows help file format.

scope A range of records selected for a given operation. Also, the "boundaries" beyond which a given variable is unavailable to another procedure or function.

scroll bar Standard window control for changing the view of data within a window, displaying more of a document or application controls than currently visible.

select To indicate to the system that the next command should act upon an on screen object, by placing the mouse cursor over it and pressing the left mouse button.

SELECT statement
(SQL) A statement setting the fields and tables for viewing, and for subsequent operations.

sequential access
The ability to manipulate all the records in a database file or table in the sequence defined by the key or index.

server A remote computer providing data storage or services to other linked computers.

SHARE.EXE The MS-DOS executable responsible for supporting multi-user access to a single file.

sort Physically rearrange all database records in a specified order, and store the results in a new database file or table.

SQL Structured Query Language; a database language for maintaining a relational database; most often-used in mainframe and client/server applications.

static text A window control which displays a string constant, and never receives focus; primarily used for labeling other controls or displaying information and instructions.

status bar An area of a window, usually found at the bottom, in which the program can display prompts and information.

stream mode A special mode for several of the Clarion database drivers which optimizes file input/output.

swap file A system file maintained by Windows for maintaining virtual memory as required by the system.

syntax A rule specifying the specific format of a language statement.

system colors The default colors shared by all custom Windows palettes.

system date The date maintained by the system clock.

table (SQL) A structured collection of data, consisting of a row of fields or column headings plus zero or more rows of data. Each row contains exactly one value for each of the fields. Within ReportWriter, the table corresponds to a specific FILE structure.

tabular report A listing of data labels and their corresponding values, arranged in a row of column labels, followed by additional rows of data arranged by column.

tag For file drivers (such as FoxPro and dBase IV) supporting multiple indexes within the same index file, the indicator marking an individual index.

text control A multi-line edit control which automatically supports word wrap.

text file An ASCII file.

text justification
A paragraph alignment style which lines up the edges of the paragraph at left, right, left and right, or centers the entire line.

third normal form
A test or measure of how closely a database meets relational theory tests for data normalization.

thumb The box control on a scroll bar.

toolbar A horizontal or vertically arranged group of command buttons, and/or other controls, generally remaining accessible the entire time a program executes.

validity check An executable code procedure which checks end user input against an expression defining acceptable values for a given field.

vector font A scalable typeface, such as a TrueType font.

vector graphic A binary file representation of a graphic or line art; defines the image as a series of lines and curves, allowing for smooth resizing. Clarion supports the .WMF (Windows Metafile) vector format.

view A virtual file containing selected fields from one or more related database files.

virtual table A data table or view which exists in memory only, constructed from one or more tables or data files which may exist on disk.

widow A portion of text or data separated from its complementary following data by a page break.

Wizard A series of dialogs that guide the user through a process, supplying defaults and limiting the user options to only those still available after each decision point, thereby controlling and simplifying the process from the user's perspective.

X axis The horizontal axis. Used for locating controls; the leftmost pixel in a window is position zero.

Y axis The vertical axis. Used for locating controls; the upper pixel in a window is position zero.

Index: