CodeReporter Help

Menus

Procedures

Output Objects

Creating/Modifying an Output Object
Deleting an Output Object
Creating a Total
Creating a Calculation
Aligning Objects
Quickly Add New Objects
Instant Layout

Groups

Creating a Group
Modifying a Group
Deleting a Group
Sizing a Group
Positioning a Group

Styles

Creating a Style

Database Functions

Creating a Relation
Specifying a Sort expression
Specifying a Query
Viewing/printing a Report

Global Settings

Global Settings
Modifying Page Header/Footer, Title, and Summary
Changing Margins

Concepts

Terms

For more information on a particular term, see the Concepts section of this manual.

Relation

Top Master Data File

Master Data File

Master Expression

Slave Tag

Slave Data File

Composite Record

Composite Data File

Complex Relation

Relation Type

Error Action

Group

Group Expression

Output Object

Total

Label

Expression

Contacting Sequiter Software

If you have a comment or question about CodeBase, or any one of Sequiter's products, please feel free to contact us by phone, fax, modem, or mail. Your comments and questions are very important to us.

Quality Control

If you are not completely satisfied with product quality or our service, please ask for, or address your comments to "Quality Control".

Technical Support

In order to help us serve you better, you must be able to provide the following information to the Technical Support Representative:

Your name, a daytime phone number, a fax number (if applicable), and the name of your company.

The fact that you are using CodeReporter.

Your CodeBase serial number. This is either on your CodeBase diskettes or the inside cover of your user's guide.

Your operating system and its exact version.

The date of the files on your CodeBase diskette. Periodically Sequiter may issue maintenance releases. The date can be determined by doing a directory listing of the CodeBase diskette.

Specify exactly what the problem is.

We are here to help and will do our best to provide high quality service.

Sequiter Software Inc.
#209, 9644 - 54 Avenue
Edmonton, Alberta, Canada
T6E 5V1
Phone: (403) 437-2410
Fax: (403) 436-2999
BBS: (403) 437-2229

Relational Reporting

Reports are used to put raw data in an understandable and organized format. They may be as simple as listing out the contents of a single data file, or as complex as the summary of the inventory of a chain of stores.

Obviously for the second, more complex report, the data would come from a number of different sources -- inventory files, part number files, description files, price files, and so on -- and would need to be combined into a single cohesive report.

It is this blending of different data files into a single report that is the basis of relational reporting. Relational reporting, put simply, is the creation of a single report using one or more data files.

See Also:

Linking Data Files
Complex Relations
Relation Types

Linking Data Files

Each data file used in a report is different and contains different data, but during the report, it is necessary for them to appear seamlessly integrated. When two or more data files are combined in a report, it is necessary to define exactly how they operate together.

In general, data files are integrated, or related, based on information common to both files. The controlling data file usually contains a reference to data that can be located in the second data file. The most common type of relation is a hierarchical Master - Slave relationship where the slave data file contains supplementary data requested by the master data file.

Data file relations are established when their interaction is exactly defined. That is, when each data file is specifically referenced, one as the controlling and one as the supplementary, and the common data is defined. When related data files are used, the data accessed in one data file causes appropriate data in the other data file(s) also to be used. The data that is "appropriate" depends on the way the relation is established.

See Also:

Master Expression and Slave Tag

Composite Record

Composite Data File

Master Data File

A master data file is a main data file that controls, or drives, another related data file. It is this data file that is used to specify which record(s) of the subservient slave data file is to be included. This is accomplished by traversing the master data file from the first record to the last, referencing data from the related data file using the established relation.

Master Expression & Slave Tag

The relation between a master and slave data file uses a dBASE expression on the master data file and an index file tag entry for the related data file. The master expression could be, for example, a field name or any supported dBASE expression (see the CodeBase 5 Reference Manual for supported expressions). This expression is evaluated for each record in the master data file and the resulting value is compared to a tag key value for the related data file. When the value is found in the index file, the corresponding record of the related data file is used.

Slave Data File

The key value obtained from the master expression is used to locate the requested data in a specific record of the related data file. This related data file is therefore used only as dictated by the master data file, and not independently, hence the term "slave" data file. Any data file that is defined in a relation and contains supplementary information is considered a slave to the master data file.

Composite Record

When a corresponding record is found in a slave data file, the two records, master and slave, are joined into a composite record which contains all the fields and data of both records. Once a relation forms a composite record, the data of the master and slave data files can be accessed as if it were in a single data file instead of two.

Composite Data File

The group of composite records created by a relation is called a composite data file. A composite data file is a term for all the records in the master data file and all related records in the slave data file. This composite file does not actually exist on disk, but is merely a specialized way of thinking about the related data files.

Complex Relations

A Complex Relation is one which has one or more slave data files acting as a master data file to another slave data file.

A master data file can be linked to more than one slave data file, and a slave data file in turn can be used as a master data file to link in yet another slave data file for further relations. This allows the creation of a relation "tree", with all the relations descending from a single top level master data file.

For example, an invoice contains three types of information: the invoice header (containing the customer's name, the date, the invoice number, etc.), the invoice lines (containing the product id, the quantity ordered, the price, etc.), and the description of the product.

The master data file is the invoice header, which has as a slave the invoice lines (using a scan relation). The invoice line does not store the product description (which could change at any time), but pulls that information from the product database.

The invoice line acts as a master data file to the product database, while acting as a slave of the invoice header database.

This "tree" effect allows the creation of highly complex reports spanning several data files.

Relation Types

There are three different types of relations, or links, between master and slave data files. Each different relation type accesses data from the slave data file differently. The three relation types available in CodeReporter are: exact match, scan, and approximate match.

When a record in a slave data file cannot be located, CodeReporter performes a specified error action .

Exact Match Relations

An exact match relation defines a one-to-one correlation between the master and slave data file. There is one -- and only one -- record in the slave data file that is accessed for the master expression. An exact match relation will always return the first corresponding record in the slave data file, even if more than one record matches the master expression.

In other words, for each record in the master data file , CodeReporter searches the slave data file until it finds a corresponding slave index tag entry with the exact same content as the evaluated master expression . At this point the search stops. In general, an exact match is best used in conjunction with a slave tag that is unique.

An Exact Match relation could be used, for example, to generate a report which displays sales orders with personal information on the salesman involved. You would establish a relation using the sales data file as the master data file and the employee data file as the slave. The databases would be related using the employee ID fields. Using an exact match in this case means that for each record in the sales data file CodeReporter will look for one corresponding record in the employee data file.

Approximate Match Relations

An approximate match relationship is similar to a exact match relation in that it has a one-to-one correspondence. The difference is that an approximate match relation returns a record whether an exact match has been found or not. When an exact match can be located it is returned. However when an exact match cannot be found, an approximate match relation returns the record *after* the closest match.

NOTE: If an exact match is not found using an approximate match relation, the record used from the slave data file is not necessarily the record closest to that of the master expression, but rather the first record greater than the master expression value

Approximate match relations do not use the error actions setting in the relation definition. Approximate match relations use the blank record error action when the evaluated master expression key value is larger than any slave tag entry.

See Figure 1.3 in the CodeReporter manual for an example using an approximate match.

Scan Relations

Scan relations define a one-to-many correspondance between the master data file and the slave data file . This means that for each record in the master data file, CodeReporter finds **all** the matching records in the slave data file, not just the first.

For example, to create a report which would display a list of all the sales made by each salesman you would establish a scan relationship using the employee data file as the master and the sales data file as the slave. The master expression would be the employee ID field from the employee data file and it would be linked to an index tag created using the employee ID field in the sales data file. Every salesman would be listed with all of his sales found in the sales data file.

Error Actions

CodeReporter allows the user to specify an action to be taken when a corresponding record is not found in the slave data file. The three supported error actions are: blank record, skip record, and stop with error.

Blank Record

When the blank record action is specified and a match cannot be found in the slave data file (using the master expression and the slave tag), CodeReporter uses blanks in place of the missing slave data.

Take the case where your employee and sales data files are related with a scan relation -- displaying the employee name and sales information. Potentially, an individual salesman may have had an off month and not listed any sales in the sales data file. If a blank record action is specified in this case, the composite record would consist of the employee data file record and a blank record from the sales data file. The employee information would be displayed, but there would be no sales information shown.

Skip Record

The skip record action also checks to see if the related slave data file contains information. When the master expression cannot be located in the slave tag, both the record in the master data file and the nonexistent record in the slave data file are ignored.

In essence, CodeReporter acts as if the record in the master data file does not exist. Using the previous example, the employee information would not be displayed if the salesman did not make a sale.

Stop With Error

The stop with error action aborts the report and displays an error message when a corresponding record in the related slave data file is not located. This action would be used when a slave data file is guarenteed to have the information related from the master data file.

For example, an accounting application could use a stop with error action when looking up critical account numbers.

Queries and Sorting

A query is a logical expression which is used to create a subset of the composite data file by specifying a search criterion. Only the records that meet the search criterion are included in the report. In additon to performing queries you can also dictate the order in which the composite records are retrieved by specifying a sort order.

See Also:

Query Expression

Sort Expression

Query Expression

A query expression is, quite simply, a dBASE expression that evaluates to a Logical value. A query expression is used to determine whether a composite record in the extended data file should be included in the report. If the query expression evaluates to .TRUE. then the record is used, otherwise it is ignored.

For example, to create a report on people who are older than 30 -- excluding the Cramford family -- the following query could be used: **PEOPLE->AGE_FIELD >30.AND.PEOPLE->L_NAME<>"Cramford"**

A person whose **AGE_FIELD** value is greater than 30 or higher would be included, while those with an **AGE_FIELD** value of 29 would not. Regardless, anyone whose **L_NAME** field starts with "Cramford" would not be included.

The Sort Expression

The sort expression is a dBASE expression which is used to determine the logical order of the composite data file . This is necessary in most cases because data in the data files is normally entered randomly and not in a sorted order. However, reports generally require that this information is displayed in alphabetic or numeric order. The sort expression in a report accomplishes this task.

When a sort expression is used, the extended data file is accessed according to the sorted ordering, and not the physical ordering of the records. Any field or combination of fields from any of the data files used in the report may be used in the sort expression.

Report Areas

A report contains several areas where information can be displayed. It should be noted that all areas are considered optional for any particular report.

Report Areas:

Title and Summary
Page Header/Footer
Groups

Title and Summary

These areas occur only once in the report; at the beginning and the end, respectively.

The title is the first area to be displayed in a report. As such, it is generally used to display descriptive information unique to the report, such as the report name. The title area can be thought of as a cover page to the report.

The summary is the last area to be displayed in a report. The summary is usually used to present final comments, an/or numerical data which summarizes the entire report.

Page Header/Footer

These areas occur once per page of the report; at the top and the bottom of the page, respectively.

Both the page header and the page footer areas are displayed on each and every page of the report; the page header area at the top of the page and the page footer at the bottom.

The page header area is generally used for items such as labels and often the page number. The page footer is often used to display running totals, page totals, and page numbers

Groups

These areas can occur anywhere in the report, and make up the bulk of the report. The group header and group footer are always added as a pair. That is, when a group is added both the header and the footer are placed in a report. There is no limit to the number of groups that may be placed in a report.

Groups are used to format the ouput of a report in an "eye pleasing" and organized manner. This is accomplished by combining common information into a single section of the report. This is best explained by way of an example.

Click here for example 1

The above example shows the contents of a composite data file. The information is sorted by month, employee ID, and product type. While this report contains all the information desired for the particular report, it is not organized well. It doesn't even list the field names for the columns.

To organize this report, composite records that have a common field value -- such as date -- should be combined under a common heading. This is done by creating a group expression.

Click here for example 2

The second example shows the same information, but with a group created for the month. This organizes the report and makes it easier to read.

Click here for example 3

The third example shows the informatino again, but with yet another, nested, group created on the employee ID.

In general, the number of groups used in a report is determined by the data, and by personal preference.

See Also:

Group Header and Footer

Group Printing and Reset Conditions

Group Header and Footer

Groups are comprised of two main parts: the group header and the group footer.

The output objects for a group may either be placed in the header or the footer. The difference between the header area and the footer area is that:

the values of the objects outputted in the header are based on the composite record that *caused* the reset condition, and

the values of the objects outputted in the footer are based on the composite record *immediately prior* to the composite record that caused the reset condition.

As groups are created they are nested within each other, that is, the header and footer of the second group defined will be between the header and footer of the first group defined.

Group Reset Conditions

When a group expression changes, a reset condition occurs. This reset causes that group's footer to be processed, its group totals and calculations to be reset, and finally the gorup's header to be processed with the next composite record.

In a multi-group report, a reset condition resets the group whose expression changed, as well as all groups between that group's header and footer.

In brief, when a group is reset:

The footers of all the groups are outputted first, before any headers are outputted. The footers are outputted in order, beginning with the innermost group, until the original group that caused the reset condition is outputted. The value of the record immediately prior to the record that caused the reset condition is used to output the group footers.

Once the footers have been processed, the headers are processed inwards, beginning with the group that was reset, using the value of the record that caused the reset condition.

Groups outside a group that encounters a reset condition are not processed.

See Figure 1.10 in the CodeReporter manual for an example on group reset condition and group printing.

01/01/1992	BK	APPLES	MACINTOSH
01/01/1992	BK	CARS	CORVETTE
01/01/1992	BK	CARS	PORSCHE
01/01/1992	BK	CARS	VOLKSWAGON
01/01/1992	SG	APPLES	FIRESIDE
01/01/1992	SG	CARS	YUGO
02/01/1992	BK	CARS	CORVETTE
02/01/1992	SG	APPLES	FIRESIDE

Products for the Month of January

BK	APPLES	MACINTOSH
BK	CARS	CORVETTE
BK	CARS	PORSCHE
BK	CARS	VOLKSWAGON
SG	APPLES	FIRESIDE
SG	CARS	YUGO

Products for the Month of February

BK	CARS	CORVETTE
SG	APPLES	FIRESIDE

Products for the Month of January

Employee ID: BK

APPLES MACINTOSH

CARS CORVETTE

CARS PORSCHE

CARS VOLKSWAGON

Employee ID: SG

APPLES FIRESIDE

CARS YUGO

Products for the Month of February

Employee ID: BK

CARS CORVETTE

Employee ID: SG

APPLES FIRESIDE

The Group Expression

Each group has an associated expression which determines when the group is displayed. For each record in the report CodeReporter evaluates this expression and stores the result. When the result of the expression changes, the new value is saved and the data for the group is displayed. If an expression is not provided for the group, the group is displayed for each record in the report.

When a groups' expression changes it is called a Group Reset Condition, or simply a reset condition. When a group resets all the groups with a lower order, in other words contained between the resetting groups header and footer, reset as well.

Output Objects

An output object is anything defined by the report designer to be displayed or printed in a report. Once the various report areas have been defined, they may be populated with output objects. An output object can be positioned anywhere within a group's header or footer area, the page header or footer, or within the title or summary area. Output objects fall into one of three categories: labels , totals , and expressions.

Labels

A label consists of a string of text which is reproduced verbatim in the report.

Expressions

Expressions are output objects which reflect the contents of the data file. An expression object is made up of a valid dBASE expression. This expression can include any of the fields in the composite data file , any predefined functions (See the dBASE Expression Appendix in the CodeBase 5 Reference Manual), and any report totals and calculations.

In general, however, expression output objects usually only contain one or more fields from the composite data file.

Totals

A total output object allows the user to summarize numerical data. Totals can be a sum, a count, a running average, a maximum value, or a minimum value. When a total object is created, it may be associated with a group. A total that is displayed in a report is made up of two parts: a total variable, and the total output object.

The total variable is a formal definition of the total that defines exactly what is totaled, and how it is maintained. The total variable may total a simple number, or one or more fields from the composite data file, or a complex expression involving fields and functions.

A total variable can be associated with a group. When this association is made, the total is reset to zero whenever the group encounters a reset condition. with the first record after the reset condition. Through this association, a total variable has the characteristics of a subtotal.

In order for the subtotal to be reset correctly, the total variable must be associated with the group immediately above the group where the total obtains its information. Omitting the group association causes the total to accumulate throughout the entire report without resetting. creates a report-wide total.

The total variable's type determines what it should do with the new data -- add it to the variable, check to see if it is the highest or lowest, increment the total by one, or use it to calculate a new average. Its type is determined when the toal variable is created. CodeReporter supports the following total types: sum , count , highest , lowest , and average .

A total variable may be outputted as a numeric value in a total output object. Alternately, a total variable may be contained in an expression output object, where it may be outputted as a numeric value or converted into a character value using a dBASE function.

A sum total adds the number returned from the evaluated total expression for each record. This value is chleared one a reset condition for the group is encountered.

A count total simply counts the number of composite records used in the group. The total expression entered for a count total is irrelevant, since every record used in the group increments this variable. is still required, but may be any valid dBASE expression -- even a zero. The expression for a count total is ignored once the total variable is created.

A highest total stores the largest number encountered for the evaluated total expression. This value is cleared once a reset condition for the associated group is encountered.

The lowest total stores the lowest number encountered for the evaluated total expression, until the associated group is reset.

Internally, an average total contains both a sum and a count total. Whenever the average total variable is needed, the sum is divided by the count, and the resulting quotient is used.

Slave Data File

The slave index tag is used in a relation to locate a record in the Slave Data File. During a report, the master expression is evaluated for each record in the master data file, and its value is used is compared against the key values stored in the selected tag in the slave index file. When the value is found in the tag, the corresponding record of the slave data file is used in the relation.

dBASE Expressions

When the term expression is used in this manual, it refers to a series of numeric, string, and/or database field values -- with or without one or more dBASE functions -- tied together in a consistent manner using dBASE operators. Expressions are consistent when all members of the expression are of the same type and are tied together using one or more dBASE operators.

See Also:

dBASE Types

dBASE Type Conversions

dBASE Operators

dBASE Types

Listed below are some of the general rules used for dBASE types. In an expression:

Any field from the composite data file is considered a variable of its field type (i.e. Character, Date, Logical, Numeric). Memo fields are considered character fields.

Anything between a pair of single (') or double (") quotes is considered a character constant. '12H3', "41", 'A' and 'CodeReporter' are all legal character constants.

Any numeric value, regardless of the number of decimal places, is considered a numeric constant. Commas, or any other separator, may not be used in a numeric value. The only separator that may be used in a numeric value is a single period denoting a decimal value.

Constant date values may not be entered directly into an expression. However, a numeric value may be added to and/or subtracted from a date value. In this case the numeric value is taken as a number of days. See dBASE Conversions for more information.

Constant logical values may be either .TRUE., .FALSE., or their shortened versions (.T., .F.). A logical value may also be generated by using one of the dBASE relational operators.

dBASE Conversions

A value of one dBASE type in an expression may be converted to another type using one of the supported dBASE conversion functions. The most common conversion functions are listed below:

DTOS(date_value). This function takes a date value and converts it to a character string in the format CCYYMMDD. A date field containing the equivalent of January 3, 1990 is converted to '19900103'.

STOD(character_value). This function takes a character value in the format of CCYYMMDD and converts it into a dBASE date value.

STR(num, len, dec). This function converts a numeric value into a character string. The len parameter is the length of the new string, including a decimal point. The dec parameter is the number of decimal places desired. All three parameters must be specified.

VAL(character_value). This function converts a character string into a numeric value. The conversion occurs from left to right and proceeds until the end of the string, or a non-numeric value, is encountered. VAL() ignores spaces. For a more complete explanation of these and other dBASE functions that may be used in an expression, see the CodeBase 5 Reference and User's Guides.

dBASE Operators

An operator is a symbol that performs an operation on two values (separated by the operator) and returns the calculated result. dBASE operators fall into four general categories: Numeric, Character, Relational, and Logical. The numeric operators (+,-,*,/,** or ^) function exactly as would algebraically be expected. Both sides of a numeric operator must be a numeric value.

Character Operators

There are two operators which may be used on character string values: + and -. Both sides of the operator must be character values.

The Concatenate I operator (+) returns a character string containing both character strings; the second appended to the first. "John "+"Smith" returns "John Smith". 'ABC'+'D' returns 'ABCD'.

The Concatenate II operator (-) returns a character string containing both character strings. However, Concatenate II inserts the second string after the last non-space character. Therefore "John "-"Smith" returns "JohnSmith ". 'ABC'-'D' returns 'ABCD'. "Spaces "-"Blanks " returns "SpacesBlanks ". The resulting string using either character operator, has a length equal to the sum of both strings concatenated.

Relational Operators

Relational operators are operators that compare two values and evaluate to a logical (.TRUE. or .FALSE.) value. The relational operators, except contain, function on all dBASE types, provided that both operands are of the same type. The Contain operator only works on character fields. The relational operators are in this table.

Logical Operators

The logical operator (.AND. and .OR.) operate on two logical operands. An additional unary operator (only needs one operand) -- .NOT. can also be used. See this table for a list of logical operators.

Equal To	=
Not Equal To	<> or #
Less Than	<
Greater Than	>
Less Than or Equal To	<=
Greater Than or Equal To	>=
Contains	$

Operation	Result
.T..AND..T.	.T.
.T..OR..T.	.T.
.T..AND..F.	.F.
.T..OR..F.	.T.
.F..AND..F.	.F.
.F..OR..F.	.F.
.NOT..T.	.F.
.NOT..F.	.T.

Calculations

A calculation performs a numerical or character-based computation that is used in the report. These computations take the form of dBASE expressions. In many cases a computation is used simply to add one or more data file fields together, but it may be more complex as necessary involving composite data file fields and/or dBASE functions.

A calculation by itself is not an actual output object. However, the computation it performs can be used as an expression or within an expression. The calculation may be thought of as a "short hand" way of referring to the computation it contains.

For example, a calculation could figure out an employee's total pay using the following formula:

```
PAY->REG_HRS*EMP->RATE +PAY->OT_HRS*EMP->RATE*1.5
```

or format an employee's first and last name using the following computation:

```
TRIM(EMP->F_NAME)+' '+EMP->L_NAME
```

If a report was simply a payroll report that listed the employees' names and gross pay, it might not be necessary to define a calculation for the formula -- the expression might just as well be put in its own output object. However, if the report also needed to display the total of all salaries paid during the report or do other computations on the employee's total pay, a calculation is suggested.

By using a calculation instead of retyping the computation in each object, the report design time is shorter, and the report is easier to modify and maintain.

Nesting

A calculation may also include one or more calculations. That is, you may nest one calculation within the definition of another calculation. For example net pay is calculated from the total pay minus any deductions. The calculation for net pay might then be:

```
TOT_EMP_PAY() - DEDUCT()
```

where `TOT_EMP_PAY()` and `DEDUCT()` are previously defined calculations.

Outputting

If a calculation is to be outputted, it must be contained within an expression output object.

Styles

A report can take advantage of the display capabilities of Windows through the use of styles. A style contains information defining a font, font size, color and special characteristics (such as bold, underline, italics, etc.) to be used in the report. A CodeReporter-defined 'PlainText' style, initially, is used as the default style for each new output object. As the report is designed, new styles may also be defined to give the report a customized look.

A style defined in CodeReporter may have attributes that can be used in both Windows and non-Windows applications. When the report is outputted in Windows, the Windows fonts are used. When outputted in a non-Windows application, the non-Windows part of the style is used.

Windows Applications

A report outputted in Windows can access the Windows drivers for color, size, font, and attribute to maintain a consistent look across different hardware combinations. When Windows does not support a particular style choice on a specific output device, Windows makes an approximation.

Non-Windows Applications

A report created with CodeReporter may also be printed or displayed by a non-Windows application that uses the report functions.

The special character attributes, such as underlined and bold letters are generally handled by the printer. The printer's special typefaces can be used by sending special printer control codes to the printer before and after an attribute is desired.

These printer-specific control codes can be embedded in a report through a CodeReporter non-Windows Style. When the CodeBase 5 report functions encounter a non-Windows style, they send the defined control codes before and after each output object that uses the style. This causes the printer to generate the desired attributes as the report is printed.

While CodeReporter saves the report's printer control codes in the report, it must be noted that each brand of printer has a unique set of control codes for each character attribute -- some may use "ESC 'B' " for bold letters, while others use "ESC 'G' ". Therefore, the stored control codes will be unique to a particular brand of printer, and may not work properly if used on another brand.

See the Skills section of this manual for creating Windows and non-Windows styles.

Each data file used in a report is different and contains different data, but during a report, it is necessary for them to appear seamlessly integrated. When two or more data files are combined in a report, it is necessary to define exactly how they operate together.

A Relation is a link between a master data file and a slave data file. The relation specifies how data in the two files is to be matched. The relation specifies a relation type and an error action.

This related data file is used only as dictated by the master data file, and not independently, hence the term 'slave' data file. This data file is controlled by the information in the master expression, and the type of relation.

A master data file is a data file that controls, or drives, another related data file. It is the master data file that is used to specify which records of the subservient slave data file are to be included. A master data file can have an unlimited number of slave data files.

The Slave Tag is an index tag which is used to locate records in the Slave Data File.

The purpose of the Master Expression is to generate an index key called the lookup key. The master expression is evaluated for each record in the Master Data File and the resulting value is that record's lookup key. A record in the Slave Data File is then found by comparing the lookup key to the Slave Tag.

When a corresponding record is found in the slave data file, the two records, master and slave, are joined into a 'composite' record which contains all the fields and data of both records. Once a relation forms a composite record, the data of the master and slave data files can be accessed as if it were in a single data file instead of two.

The group of composite records created by a relation is called a Composite Data File. A composite data file is a term for all the records in the master data file and allrelated records in the slave data file(s). This composite data file does not actually exist on disk, but is merely a specialized way of thinking about the related data files.

A Complex Relation is a relation which has one or more slave data files which act as a master data file to another slave data file. A master data file can be linked to more than one slave data file, and a slave data file in turn can be used as a master data file to link in yet another slave data file for further relations.

This allows the creation of a relation tree, with all the relations descending from a single top level master data file.

An Exact Match relation permits only a single match between the master and slave data files. In other words there is a one-to-one correlation between the master and slave. If there are multiple slave records that match a single master record, a composite record is only generated for the first matching slave record.

An Approximate Match relation is similar to an exact match relation in that it has a one-to-one correspondance. The difference is that an approximate match relation will always find a corresponding record, even if the length or content of the slave tag is not exactly the same as the master expression.

As CodeRepoter searches through the slave tag it keeps track of the tag value. As soon as the tag value is equal to or greater than the master expression the search stops and the current slave record is used.

Scan Relations define a one-to-many correspondance between the master and slave data files. This means that for each record in the master data file CodeReporter finds all the matching records in the slave data file, not just the first.

Each relation has a specified relation type which determines the manner in which the two data files are related. The available relation types are: exact match, approximate match, and scan.

Each relation has a specified error action which determines what action to take when no corresponding record is found in the slave data file. The available error actions are: blank record, skip record, and stop with error.

When the Blank Record error action is specified and a match cannot be found in the slave data file, CodeReporter uses blanks in place of the missing slave data.

The Skip Record action causes records in the master (and slave) data file to be ignored if there is no corresponding record in the slave data file.

The Stop With Error error action aborts the report and displays an error message when a corresponding record in the related slave data file is not found.

A group is a collection of similar data combined together as a set in a report. Groups are used to display statistics on records that have something in common. The statistics displayed can inlcude fields from the composite record, totals, calculated values, etc.

See Group Concepts for more information.

Each group in a report has an associated expresion which determines when the group is displayed. For each record in the report the group expression is evaluated and the result is stored. When the result of the group expression changes from one record to the next the group displays its data and resets itself. If no expression is provided the group resets for each record in the report.

An output object is anything defined by the report designer to be displayed or printed in a report.

Once the groups of a report have been defined they may be populated with output objects. An output object can be positioned anywhere within a group's header or footer area, the page header or footer, or within the title or summary area.

Output objects fall into one of three categories: labels , expressions , totals.

A label is a string of text which is reproduced verbatim in the report.

Expressions are objects which reflect the contents of the data file. An expression object is made up of a valid dBASE expression , which can include any predefined functions (See the CodeBase 5.0 manual), any fields from any data file in the report relation, and any report totals and calculations.

A total object allows the user to summarize numerical data. Total objects may only display previously defined total variables. Total variables can be a sum, a count, a running average, a maximum value, or a minimum value.

When a total object is created it may be associated with a group. A total object that is associated with a group will be reset to zero whenever the group is reset. If the total object is not associated with a group the total does not reset at any point in the report.

The term Top Master Data File refers to the top data file in the relation tree (the main data file of the report). It is this data file that is specified when the report is initially created.

Sort Expression

To set a sort expression for the composite data file select Sort from the Database menu. When the Expression Entry dialog appears enter a valid dBASE expression for CodeReporter to use in sorting the composite records returned by the relation.

Query Expression

To set a query expression for the composite data file select Query from the Database menu. When the Expression Entry dialog appears enter a valid dBASE expression for CodeReporter to use in filtering the composite records returned by the relation.

The query expression must evaluate to either .TRUE. or .FALSE.. The query expression is evaluated for each record in the relation. If the result is .TRUE. the record is included in the composite data file.

Creating A Total

Creating a Total type output object varies slightly from creating and Expression or Label. In order to create the output object a total must first be defined.

Defining A Total

To define a total select the Report | Totals menu option to invoke the 'Totals' dialog box.

To create a total variable, select the 'Create' button. CodeReporter prompts for a unique name for the total.

When a name has been provided the Expression Entry dialog appears. Use the dialog to enter an expression for the total. This total expression is evaluated for each composite record used in the report and the resulting value is used to keep the total current. The expression often consists simply of one of the fields in the data file, but can be more complex and even be dependent on the contents of various fields.

Select a reset group for the total by clicking on the desired group in the list of groups shown in the 'Reset Group' list box. A total without a reset group is never reset and is, in effect, a report wide total.

Finally select the desired type from the 'Type' list box (sum , count , highest , lowest , average.

Creating The Total Output Object

Once a total variable is defined, a Total output object can be created by using the 'Create Object' dialog box. (See Creating Output Objects). Once invoked, select the 'Total Lookup' button, and select the desired total variable from the displayed list.

More than one output object can use the same defined total, and a defined total can also be used in the expression for an expression object or another total.

A total variable may be used in an expression by entering the total variable's name in UPPERCASE LETTERS, followed by parentheses.

Creating A Calculation

A calculation is similar to a total in that it must be defined before it can be used in an output object. Unlike a total there is no output object specifically for calculations. Instead a calculation is used as the expression, or part of the expression, for an expression type output object.

Defining A Calculation

To define a calculation select the Report | Calculations menu option to invoke the 'Calculations' dialog box.

To create a calculation select the 'Create' button and when prompted enter a unique name for the calculation. When the Expression Entry dialog appears enter an expression for the calculation. This can be any valid dBASE expression which evaluates to a numeric value.

Modifying A Calculation

To edit a calculation's expression select the desired calculation name in the 'Name' list box and change the expression in the 'Expression' edit control. Alternatively select the desired calculation and click on the Easy Expr. button. This invokes the Expression Entry dialog.

Deleting A Calculation

The 'Delete' button can be used to delete the currently selected calculation.

Caution! Deleting a calculation automatically removes any output objects, calculations, and totals that use the calculation. Remove calculations with care.

Renaming A Calculation

The 'Rename' button can be used to change the name of the currently selected calculation. The name is automatically changed througout the report.

Creating An Output Object

All output objects are created using the 'Create Object' dialog box. This dialog may be invoked in three ways: using the mouse, using the keyboard, or using a menu item.

Using The Mouse

To create an output object using the mouse simply position the mouse cursor at the point where the new object is desired and click on the right mouse button. This brings up the Create Object dialog. Enter the appropriate text for the output object and set the various options presented in the dialog and then click on OK. The new object will be positioned with its upper left corner at the point where the right mouse button was clicked.

Using The Keyboard

To create an output object using the keyboard press the insert key. The Create Object dialog will be brought up allowing for specification of the output object. The output object is created at location zero, zero in the currently selected report area.

Using The Menu

To create an ouput object using the menu select the New Object item from the Object menu. This has the same effect as pressing the insert key. A new output object is created at zero, zero in the currently selected report area.

Deleting An Output Object

To delete an output object select the object to be deleted, either by clicking on it with the mouse, or by tabbing to it using the tab key. Once the desired object has been selected, simply press the delete key or use the Objects | Delete menu option.

Viewing/Printing A Report

Viewing The Report

To view a report select the Display menu item from the File menu. The report will be displayed in a viewing window. Use the Next Page menu item to advance through the report, and the Close Window menu item to leave the view window.

Printing The Report

To print a report select the Print item from the File menu. The report will be printed on the printer selected in the Printer Setup dialog. To choose the printer to be used select the setup printer item from the files menu and choose the appropriate printer from the displayed list.

Page Header/Footer, Title and Summary

To modify the size of the Page Header/Footer, Title and/or Summary report areas, select the Groups | Titles, Headers, Footers, Summary menu option. This option invokes the 'Page Header/Footer, Title and Summary' dialog box where the sizes may be modified.

Initially, these areas have a size of zero -- indicating that they are not used in the report. In order to place output objects in these areas, they must have a positive size. Enter the desired size of these report areas in their respective edit controls and select 'OK' to save the sizes.

File Menu

Selecting Open calls up a dialog box which allows the user to select a previously saved report for retrieval. It also allows the user to choose to create a new report.

Selecting New allows the creation of a new report.

Selecting Save allows the current report to be saved. If the current report has not yet been saved a dialog box prompts for a name under which to save the current report. Otherwise the current report is saved with the existing name.

Save As calls up a dialog box prompting the user for a name under which to save the current report.

Selecting Save As Code allows the report to be saved as a C source code file. The source code uses CodeBase 5.0 function calls to generated the defined report.

Selecting Display causes the report to be generated and displayed on the screen. The user can then page through the report.

The Print option generates the defined report and sends the output to the selected printer.

Printer Setup allows the selection of the printer to be used for output.

Exit leaves the application the application. If the report has changed since the last save, or has not yet been saved, the user is prompted to save the file.

Style Menu

The style menu is used to create, delete, and modify the styles used with output objects. The style can be specified for Windows applications, Non-Windows applications, or both.

Windows Styles

To create a new style under Windows, select the Styles menu option to invoke the 'Styles' dialog box. (See Dialog Summary in the CodeReporter manual for more information on the 'Styles' dialog)

Select the 'Create' button, and enter a descriptive name for the new style. The newly created style uses the attributesof the currently selected style as its default values. The style may then be made unique by choosing the 'Modify' button and changing the attributes of the style.

The 'Modify' button invokes the 'Fonts' dialog. Select the desired font, font size, color, etc. and click the 'OK' button.

See Selecting a Style for more information on using a style with an output object.

Non-Windows Styles

To create a new style, select the Styles menu option to invoke the 'Styles' dialog box. (See Dialog Summary in the CodeReporter manual for more information on the 'Styles' dialog)

CodeReporter's Non-Windows styles use a printer's built in capabilities to create the different typefaces used in a report. As such, it is necessary to obtain the printer's documentation to determine exactly what typefaces are available for any printer.

Once the 'Styles' dialog is invoked, select the 'Create' button, and enter a descriptive name for the style. The new style creates a copy of the currently selected style. To make the non-Windows style unique, select the 'Non-Windows' button and enter in the hexadecimal pre-string and post-string control codes from the printer's documentation.

The codes are entered in the format \xhh\xhh where *hh* represents a two digit hexadecimal number. If more than two characters are necessary in the printer control code, they may be appended, without a space, to the end of the control string.

See Selecting a Style for more information on using a style with an output object.

Selecting a Style

An output object is automatically assigned the default style when it is created. The styles of output objects may be changed on an object-by-object basis by invoking the 'Modify Object' dialog box for each output object.

Once invoked, the 'Styles' list box may be used to select a previously defined style. Select the 'OK' button to save the changes, or 'Cancel' to abort the selection of the new style.

Database Menu

Relations allows the user to to form a multiple database relation.

Top Master Tag allows the user to select a tag on which the master data file is indexed. This tag is ignored if a sort expression has been specified.

Sort is used to enter a sort expression. The sort expression can contain references to any of the databases in the current relation.

Database Relations

CodeReporter allows the user to specify a multiple data file relation. This creates a compound record containing the appropriate record data from each data file.

A relation consists of a master database, and any number of slave data files. Each of the slave data files can be a master to other slave data files, in effect creating a relation tree.

A data file may be used only once in a relation. In order to link a master and slave data file a linking field, or expression, is set for the master data file. This field, or expression, must have a value which matches a value in the slave data file tag. An index into the slave data file must be selected. The value of this index is compared to the master expression.

A relation between a master and slave data file can have one of the three relation types. These types are exact match, approximate match, and scan.

In addition the error action for the relation can be specified. This is the action taken when a corresponding record in the slave data file is not found.

Creating a Relation

A relation tree can be initiated by selecting the Database | Relations menu option. When the Relations dialog box first appears, it has no relations defined, and the top master data file is displayed in the upper left corner.

See Also:

Adding a Slave

Deleting a Slave

Match Length

Traversing a Relation

Relation Types

Error Actions

Traversing a Relation

By double clicking on one of the files in the slave list box, the selected slave becomes the current master.

Alternatively the 'Down', 'Up', and 'Next' buttons can be used. The 'Down' button moves down the relation tree, using the first slave data file of the current master for the new current master. The 'Up' button works similarly but moving in the opposite direction.

The 'Next' button provides a simple, organized way of moving through large and complex relation trees. This is done by traversing the relation tree from top to bottom and left to right for each press of the button. The advantage of using the 'Next' button instead the 'Up' and 'Down' buttons is that a complete traversal of the tree can be done by clicking one button, instead of several 'Up', 'Down', and arrow keys.

The 'Next' button traverses the relation tree by moving left to right and going down the relation tree whenever possible. See Figure 2.6 in the CodeReporter manual for an illustration of this technique.

Adding a Slave

To add a slave to the current master click on the 'Add Slave' button. The 'Choose Slave Data File' dialog appears. Use the Files, Directories, and Drives list boxes to select the desired slave data file. When the desired file has been selected click 'OK'.

Once the slave data file has been selected, the 'Choose Slave Index File' dialog appears. Use this dialog to select the index file which contains the slave tag. If the index file contains more than one tag, the desired tag can be selected by using the Index Tag combobox in the Relations dialog. Additionally a new index file can be opened for an already existing relation by clicking on the 'Open New Index' button in the Relations dialog.

After the slave data file has been set the Expression Entry dialog appears. Use this to create the master expression. The master expression for an existing relation can be modified by direct editing with the 'Master Expression' edit control in the Relations dialog, or by using the 'Easy Expr.' button.

Adding a Slave to a Slave

A slave may be used as a master to another data file. To do this, double click on the name of a slave data file from the 'Slave List' list box. (Selecting 'Down' button while the slave's name is highlighted accomplishes the same task.) This slave then becomes the current master data file and its name is displayed in the upper left corner of the 'Relations' dialog box. Select the 'Add Slave' button and follow the same procedures listed above on adding a slave.

When the new master-slave relation is complete, select the 'OK' button, or continue to add new relations.

Deleting a Slave

A slave may be deleted from the relation by highlighting the slave data file's name in the 'Slave List' list box (in the 'Relations' dialog) and selecting the 'Delete Slave' button.

WARNING! Removing a slave from the relation tree also removes any lower level relations based on the deleted slave and any output objects, calculations, and totals that use these data files. Remove a slave with care.

The 'Match Length' edit control value should be changed if the length of the master expression is greater than the slave tag entry, or if a smaller portion of the master expression and slave tag entry is to be used in the relation.

The match length is ignored when master expressions evaluate to numeric values.

Print Menu

Go Sends the report to the currently selected output device. Options Allows the user to select a printer for the report output. Optionally checking the 'Display Report' check box causes the report to be sent to the screen.

Report Menu

Totals lets the user to create, edit, and delete totals objects. These totals can then be incorporated into the report as output objects. A total can be a sum, a count, a lowest value, a highest value, or a sum. Each total is attached to a reset group. The sum, count, etc. applies to all the records in this group and each time the group resets so does the total. If the reset group is left blank the total continues throughout the report. This is often useful in generating summary information.

Calculations allows the user to create, edit, and delete calculation objects. These calculations can then be incorporated into the report as output objects.

Query Allows the user to enter a query expression. This expression may contain references to any database in the current relation. Only records which meet the criteria of the query statement will be included in the report.

Global Change Allows the user to adjust the report size and margins, the default date format, and the default units for the report.

Instant layout text.

Add new objects text.

Alignment Menu

It is often convenient to have many output objects in one or more groups aligned in the same manner. By selecting multiple objects and using the Align menu options, they can quickly be aligned along the right, left, center or top of the first object selected.

As an added convenience, when objects are aligned using the Right, Left, or Center menu options, the objects' rectangles are not only aligned with the other objects, but the outputted text is also justified in a corresponding manner. That is, an output object that is aligned on the right border of another output object (using Align | Right) will be outputted with

Left Align allows the user to left align a group of output objects.

Right Justify allows the user to right align a group of output objects.

Center Justify allows the user to center a group of output objects.

Vertical Move allows the user to vertically align a group of output objects. The justification of the output object(s) selected does not change.

Sensitivity Allows the user to set the vertical and horizontal sensitivity of object movement.

Aligning Objects

To align a group of objects first select the objects to be aligned by using the selecting multiple objects procedure. Once the objects to be alignmed have been selected choose one of the alignment options from the Align menu. The selected objects will then be aligned with whichever object was selected first.

Right Aligned Objects , Left Aligned Objects , Center Aligned Objects , Vertically Moved Objects .

Objects right aligned using the 'Student' as the first selected output object.

Student

Job

Age

Student

Job

Age

Objects center aligned using the 'Student' as the first selected output object.

Student

Job

Age

Student

Job

Age

Objects left aligned using the 'Student' as the first selected output object.

Student

Job

Age

Student

Job

Age

Objects left aligned using the 'Student' as the first selected output object.

Student

Job

Age

Student Job Age

Selecting Multiple Objects

To select a group of objects click on the first object with the left mouse button. Then hold down the shift key while clicking on the other items to be selected. A group of selected items can be moved as a group, or can be aligned using the Align menu.

Group Menu

New Group allows the creation of a new group.

Modify Group allows the user to modify the header/footer size, position, name, etc. for the currently selected group.

Delete Group deletes the currently selected group.

Size Groups allows the user to adjust the header and footer size of all the created groups.

Position Groups allows the user to adjust the position of any of the created groups.

Title, Headers, Footers, Summary. This menu option allows changes to the size of the page header ,page footer, title and summary . Initially these areas have a zero size. By changing the size of these areas, output objects may be added to them.

Creating A Group

To create a group select Group | Create Group menu option. When prompted, enter a name for the new group. Once created, the 'Modify Group' dialog box is invoked to allow custom changes to the group.

The newly created group is placed between the header and footer of the last created group and has the highest position number. To change the group's position use the Modify Group dialog or the Groups | Position Groups. menu option.

By default the group is created with a header size of 24 points and a footer size of zero points. To change the header and/or footer size use Modify Group or Size Groups.

Swap Header

If a group uses the 'Swap Header' option and it encounters a reset condition, a new page is generated and the group's header is outputted in the position of, and instead of, the page header. Since this generates a new page, it is uncommon to have more than one group swap their headers.

In the same manner, if the 'Swap Footer' option is used and a reset condition ccurs, the group footer is used in the position of, and instead of, the page footer.

The 'Swap Header' option is generally used to display special information such as an invoice header, a different format or font for existing information, or a cover page.

See the CodeReporter manual for a detailed explanation of the 'Swap Header' and 'Swap Footer' options.

Repeat Header

The 'Repeat Header' option is used to display the group header at the top of a new page when the end of page occurs before a group reset condition is encountered. This is particularly useful for maintaining column titles across page boundries.

Reset Page Number

The current page number is reset to '1' for every group reset condition encountered when the 'Reset Page #' option is selected for the group.

Reset Page

With the 'Reset Page' option enabled, the following sequence of events occur when a group reset condition occurs:

The group and page footers are outputted, and then a form feed is sent to the printer.

The next page header and the current group's header are then both processed normally at the beginning of the new page.

Modifying A Group

To modify a group select the group to be modified by clicking inside either the header or footer area with the left mouse button. Once the group has been selected choose Modify Group from the Group menu. The Group Modify dialog will appear.

The group modify dialog allows the user to change the group's name , position , header and footer height , group expression , as well as set the swap header , swap footer, reset page , reset page number , and repeat header flags.

Group Name

A unique group name is assigned when a group is created. It is this descriptive name that identifies the group area in all subsequent dialog boxes as well as in the report design screen.

The name of a group may be modified from the 'Modify Group' dialog box. When invoked, the 'Name' combo box contains the name of the currently selected group. Changing the name in this combo box changes the name of the group throughout the report.

Deleting A Group

To delete a group select the group to be deleted by clicking inside either the header or footer area with the left mouse button. Once the group has been selected choose Delete Group from the Group menu.

Caution: When a group is deleted any objects it contains are also deleted.

Sizing A Group

Changing the size of a groups header or footer can be accomplished in two different ways. The first is to use the Modify Group member of the Group menu. This invokes the 'Modify Group' dialog box where the explicit size of the group may be set.

The second is to use Size Groups from the group menu.

When Size Groups is selected from the Group menu the Size Groups dialog appears. This dialog allows the user to modify the header and footer size of any of the created groups. See the Dialog Summary in the CodeReporter Manual for more information on the 'Size Groups' dialog.

Positioning A Group

A groups position can be changed in two different ways. The first is to use the Group | Modify Group menu option. This invokes the 'Modify Group' dialog box. The 'Position' edit control determines where the group is outputted in relation to the other groups added. Changing the 'Position' edit control changes the group's position.

The second is to use the Groups | Position Groups menu option. When Position Groups is selected, the 'Position Groups' dialog appears. This dialog allows the user to modify the position of any of the created groups.

Page Header/Footer, Title and Summary

Page Headers, etc.

Page headers, and page footers are special instances of groups. The page header is output at the top of each page and the page footer is output at the bottom of each page.

The Title and Summary are also special instances of groups. The title is, as its name implies, printed once at the beginning of the report. The summary is printed at the end of the report. Often the summary is used to contain totals which ran throughout the report.

Output Objects are placed in the page header/footer, title, and summary in the same manner as in any other group.

Global Settings

CodeReporter default and global settings are accessed using the Report | Global Settings menu option. These settings provide an initial set of values for every report.

See Also:

Default Date Format

Decimal Point Character

Thousands Separator Character

Currency Symbol

Unit of Measurement

Margins

Report Width

Default Style

Hide Object List Info

Use Leading Zero

Default Date Format All output objects that evaluate to a date value are outputted using these formatting characters. Changing this format setting changes the way all dates are outputted. Date output objects that have been formatted individually ignore this setting.

When CodeReporter outputs fractional numbers, it places this character between the whole number and the fractional part. In most cases, the decimal point is simply a period. However, in some countries commas are used as a separator. Changing this value changes the way fractional numbers are outputted.

Thousands Separator Numeric values greater than one thousand may use a character placeholder to format the output to the thousand, million, billion, etc. place. The comma is used as the default separator. However, it may be changed to any character, a space, or be deleted completely. If the 'Thousands Separator' is specified, all numeric values use it as a placeholder. If it is deleted, the numbers are displayed in raw form.

Using a comma separator	1,000,000
Using no separator	1000000

When a numeric value is formatted as a currency value, the 'Currency Symbol' character (found in the 'Global Settings' dialog) is placed before the outputted value.

The default currency is for Dollars ($). However, by changing the value in the 'Currency Symbol' edit control, any character may be outputted. Listed herein are some common currency symbols supported by the standard Windows character set.

Name	Key Strokes
Cent	Alt+0162
Pound	Alt+0163
Yen	Alt+0165

The unit of measurement that CodeReporter uses to display position and size information are determined by the 'Units' list box (found in the 'Global Settngs' dialog). Modifying the unit of measurement does not effect the position or size of anything in the report.

Margins

The distance the report is offset from the top, bottom, left, and right edges of the printed page is controlled using the 'Left Margin', 'Right Margin', 'Top Margin', and 'Bottom Margin' edit controls in the 'Global Settings' dialog box.

These settings not only effect the printed page, but also the report design screen. If a left or right margin is changed, the report design screen shrinks or expands to the appropriate size.

See Also:

Report Width

The 'Report Width' edit control changes the overall horizontal size of the report's page. This value is initially set to the current page size of the default printer. A report may be designed for a wider page than that for which the selected print is configured.

If the 'Report Width' setting is larger than the printer's page when printing in Windows, the report is truncated to the appropriate size. Otherwise, the report may not be outputted correctly.

The page settings are based on the current configuration of the selected output device. If these settings are incorrect, or if a special size of form is used in the printer, make the necessary changes to the printer's setup. These changes can be made using the Printers option in the Windows Control Panel.

See the *Windows User's Guide* for information on changing a printer's settings.

All new output objects, by default, are created with the style selected in the 'Default Style' list box of the 'Global Settings' dialog box.

Changing the default style does not effect any object already created, but does change the style of any new object added.

CodeReporter uses the 'PlainText' style as a default until another default style is selected.

The object list information areas in the report design screen may be hidden by clicking the 'Hide Object List Info' check box of the 'Global Settings' dialog box. They may be made visible by clicking the check box again.

Fractional numbers (those between 1 and -1) are represented by the decimal point character and the fractional number. In some reports, it may be desirable to have a zero placed before these types of numbers. Selecting the 'Use Leading Zero' check box of the 'Global Settings' dialog box causes CodeReporter to place a zero before the decimal point character.

Use Leading Zero	Do not Use Leading Zero
0.3	.3
33.3	33.3
-0.3	-.3

Quickly Add New Objects

CodeReporter provides a second way of adding output objects. Using the Report | Quickly Add New Objects menu option new output objects can be placed with a minimum amount of work.

Several output objects can be created and placed to the right and/or downwards of each other, with a specified offset between them, with a couple clicks of the mouse.

Once the objects are quickly created, they are exactly the same as any other object and may be moved, modified, and deleted in the same manner.

Using Quickly Add New Objects

Select the Group The 'For New Object' list box displays all of the previously created groups wherein objects may be placed. Objects created with the 'Quickly Add New Objects' dialog box are placed in the selected group.

When the group changes, the position edit controls also change to reflect the position in the group CodeReporter will place the new objects. Select the Field or Expression Many output objects are simply a field from the composite data file. By selecting one of the fields in the 'Fields' list box and selecting the 'Create' button, an output object is created for that field. If a more complex expression is desired, select the 'Easy Expr.' button and enter the expression with the Expression Entry dialog box.

Add a label If a label output object is used to describe the new output object, it also may be created at the same time as the output object. When the 'Create' button is selected with the 'Add Label' box checked, a label object is also created. The new label object contains the text in the 'Label Text' edit control, and is placed in the group highlighted in the 'For New Label' list box.

A label may be offset from the object it describes by changing the position specified in the edit controls under 'Label Offset'. Change Positions The suggested position of the next object created is shown in the lower left edit controls. The object position values are automatically updated each time an object in the selected group is created. The object is positioned according to the coordinates displayed under 'Object'.

Once created, the values under 'Offset' are added to those under 'Object' to obtain the position of the next object. These values are maintained individually for each group, so that objects may be added to groups in a random order, while still keeping an organized report. An object cannot be placed so that it extends beyond the report width or outside of the selected report area.

Instant Layout

The Report | Instant Layout menu option creates an instant report containing every field in the composite data file with a page header containing the field names.

The Instant Layout option is a very quick way of creating a labeled "data file dump" report. Once Instant Layout has placed the output objects, they may be modified and/or deleted in the same manner as any other output object. Instant Layout places as many of the composite data file fields as can fit to the width of the report (see Global Settings). Once a field cannot fit into the remaining width, Instant Layout stops placing output objects.