APPENDIX

Formula Language Reference

This appendix provides detailed information for each Crystal Reports formula language function and operator. Functions and operators are categorized by usage (string, numeric, financial, and so forth). The purpose of each function or operator is discussed, along with explanations of various function arguments. An example of each function and operator is provided.

Consider the following points when making use of this reference:

- Crystal Syntax is used in this reference. With very few exceptions, Basic Syntax is not covered, as Visual Basic and Visual Basic for Applications (which Basic Syntax is based on) are commonly documented languages. In many cases, the Basic Syntax version of the function or operator is identical or very similar to the Crystal Syntax version documented here.
- Some more esoteric functions, such as statistical and financial functions, may not be explained in great detail. It is assumed that report designers who need to use these specialized functions will already possess a basic understanding of their usage.
- Although most of the examples in this appendix use "literal" arguments, you may easily substitute database fields, other formulas, and so forth, for function arguments.
- Formatting in examples that follow each function or operator assume standard U.S. settings in the Windows control panel. If you've changed these settings, your default output formatting may be different.

Functions: Additional Functions

This category of the Function tree exposes functions that are provided to Crystal Reports by a set of external dynamic link libraries (DLLs). These DLLs, known as User Function Libraries (UFLs) within Crystal Reports, can be developed in a Windows programming language to provide extensible functions to the Crystal Reports formula language. UFL functions are categorized by their UFL DLL filename.

NOTE *If you have upgraded from previous versions of Crystal Reports, or you have downloaded additional User Function Libraries from Business Objects (or other sources), you'll find additional entries in the Additional Functions category of the Function tree.*

ByteToText

Converts a number to a descriptive string indicating disk or memory storage used.

ByteToText is useful when reporting on the Local File System (or other data source that requires amounts of computer storage to be included on the report). Values less than 1,204 are returned in bytes, values between 1,204 and 1,048,576 are abbreviated to "KB", and values above 1,048,576 are abbreviated to "MB".

ByteToText (n)

n – numeric value.

```
ByteToText(1000)
```

returns "1000 bytes".

```
ByteToText(10000000)
```

returns "9 MB".

DateTimeTo2000

Converts a date-time value that may not be year 2000–compliant to a year 2000–compliant date-time with a four-digit year.

DateTimeTo2000 is "left over" from previous versions of Crystal Reports that may not have dealt with date-time values properly. It is largely not required with current versions, unless you may be dealing with older databases that don't return full four-digit years in date-time fields.

DateTimeTo2000 (dt, n)

dt – date-time value with a two- or four-digit year.
n – numeric value to act as a "sliding scale" year window.

```
DateTimeTo2000(DateTimeValue("1/1/90 1:03 pm"), 95)
```

returns 1/1/2090 1:03:00 PM.

DateTimeToDate

Returns only the date portion of a date-time value.

This function is "left over" from previous versions of Crystal Reports and has been replaced by the CDate and DateValue functions. See **DateValue** for details.

DateTimeToSeconds

Extracts the number of seconds that have passed since midnight.

DateTimeToSeconds evaluates the time portion of the supplied date-time value and returns the number of seconds that have passed since midnight.

DateTimeToSeconds (dt)

dt – date-time value.

```
DateTimeToSeconds(CurrentDateTime)
```

returns 49,980 for a current date-time value that includes the time 1:53:00 P.M.

DateTimeToTime

Returns only the time portion of a date-time value.

This function is "left over" from previous versions of Crystal Reports and has been replaced by the CTime and TimeValue functions. See **TimeValue** for details.

DateTo2000

Converts a date value that may not be year 2000–compliant to a year 2000–compliant date with a four-digit year.

DateTo2000 is "left over" from previous versions of Crystal Reports that may not have dealt with date values properly. It is largely not required with current versions, unless you may be dealing with older databases that don't return full four-digit years in date fields.

DateTo2000 (d, n)

d – date value with a two- or four-digit year.
n – numeric value to act as a "sliding scale" year window.

```
DateTo2000(#1/1/90#, 95)
```

returns 1/1/2090.

DTSTo2000

Returns a string value from a date string or date-time string with a two- or four-digit year.

DTSTo2000 will convert a string formatted in a specific year-month-day-hour-minute-second format to another string with a four-digit year. This function includes a sliding-scale year argument to ready two-digit dates for year-2000 compliance.

DTSTo2000 (s, n)

s – string containing a date in the format "yyyy/mm/dd" or "yy/mm/dd" or date-time in the format "yyyy/mm/dd hh:mm:ss.00" or "yy/mm/dd hh:mm:ss.00".
n – numeric value to act as a "sliding scale" year window.

```
DTSTo2000("90/10/01",95)
```

returns the string "2090/10/01".

DTSToDate

Converts a string formatted in a specific date-time format to a true date value.

DTSToDate is functionally equivalent to CDate and DateValue. See **DateValue** for more information.

DTSToDateTime

Converts a string formatted in a specific date-time format to a true date-time value.

DTSToDateTime is functionally equivalent to CDateTime and DateTimeValue. See **DateTimeValue** for more information.

DTSToSeconds

Extracts the number of seconds that have passed since midnight.

DTSToSeconds evaluates the time portion of the supplied date-time string and returns the number of seconds that have passed since midnight.

DTSToSeconds (s)

s – string containing a date-time in the format "yyyy/mm/dd hh:mm:ss.00" or "yy/mm/dd hh:mm:ss.00".

```
DTSToSeconds("2003/05/28 13:53:00.00")
```

returns 49,980 (the number of seconds between midnight and 1:53 P.M.).

DTSToTimeField

Converts a string formatted in a specific date-time format to a true time value.

DTSToTimeField is functionally equivalent to CTime and TimeValue. See **TimeValue** for more information.

DTSToTimeString

Returns a string containing just the time portion of a date-time string.

DTSToTimeString simply extracts the text after the eleventh character of a string (which does not actually have to be formatted as a date-time string).

DTSToTimeString (s)

s – string containing a date-time in the format "yyyy/mm/dd hh:mm:ss.00".

```
DTSToTimeString("1998/05/28 13:53:00.00")
```

returns the string "13:53:00.00".

NOTE *The preceding series of date-time string (DTS) functions are "left over" from previous versions of Crystal Reports that may not have dealt with date-time values properly. For example, Crystal Reports 6 did not properly recognize a true date or date-time value from a database but returned these values as string fields in a certain format. While you can still make the choice to convert date or date-time values to strings, these DTS functions are generally not required with current Crystal Reports versions.*

EventNumber

Returns a text description of a Microsoft Exchange event number.

EventNumber is helpful when reporting from Microsoft Exchange data sources. Certain supplied event numbers are significant (for example, 1,000 indicates local delivery). Those that aren't recognized will return "unknown event" results.

EventNumber (n)

n – numeric value representing a Microsoft Exchange event number.

```
EventNumber(1000)
```

returns "Local Delivery".

ExchGetId

Extracts the e-mail address from an X400- or X500-formatted string.

ExchGetID is helpful when reporting from Microsoft Exchange data sources. These data sources return certain e-mail address strings that can be passed to ExchGetID to return just the e-mail address.

ExchGetId (s)

s – string containing an X400- or X500-formatted address.

```
ExchGetID("c=US; a= ;p=Ablaze Group;
o=AblazeServer;dda:smtp=Info@AblazeGroup.com")
```

returns "Author@CrystalBook.com".

ExchGetOrganization

Extracts the organization string from an X400- or X500-formatted string.

ExchGetOrganization is helpful when reporting from Microsoft Exchange data sources. These data sources return certain organization strings that can be passed to ExchGetOrganization to return just the organization.

ExchGetOrganization (s)

s – string containing an X400- or X500-formatted address.

```
ExchGetOrganization("c=US; a= ;p=Ablaze Group;
o=AblazeServer;dda:smtp=Info@AblazeGroup.com")
```

returns "Ablaze Group".

ExchGetPath

Extracts the destination path string from an X400- or X500-formatted string.

ExchGetPath is helpful when reporting from Microsoft Exchange data sources. These data sources return certain destination strings that can be passed to ExchGetPath to return just the destination path.

ExchGetPath (s)

s – string containing an X400- or X500-formatted address.

```
ExchGetPath("c=US;a= ;p=Ablaze Group;ou1=Info;
ou2=Ablaze Group;o=AblazeServer;dda:smtp=Info@AblazeGroup.com")
```

will return "OU1=INFO; OU2=ABLAZE GROUP;".

ExchGetSite

Extracts the site string from an X400- or X500-formatted string.

ExchGetSite is helpful when reporting from Microsoft Exchange data sources. These data sources return certain site strings that can be passed to ExchGetSite to return just the site.

ExchGetSite (s)

s – string containing an X400- or X500-formatted address.

```
ExchGetSite("c=US;a= ;p=Ablaze Group;ou1=Info;ou2=Ablaze Group;
o=AblazeServer;dda:smtp=Info@AblazeGroup.com")
```

will return "AblazeServer".

ExtractString

Extracts the portion of a string that appears between two other strings.

ExtractString will search a string, returning the portion that falls between two other search strings. If the first search string is not found, ExtractString returns an empty string. If the second search string is not found, ExtractString will return the entire portion of the source string after the first search string.

ExtractString (s1, s2, s3)

s1 – source string to be searched.
s2 – first string to search for.
s3 – second string to search for.

```
ExtractString("The rain in Spain falls on the plain",
"rain", "plain")
```

returns "in Spain falls on the".

FRAccRecTurnover

Returns the average turnover of accounts receivable in numeric days.

FRAccRecTurnover calculates the average "turnover" of accounts receivable by dividing accounts receivable by sales, and multiplying the result by the number of days in the year.

FRAccRecTurnover (n1, n2, n3)

n1 – numeric value of total accounts receivable.
n2 – numeric value of total sales.
n3 – numeric value indicating the number of days in a year (default is 360).

```
FRAccRecTurnover(10000,100000,360)
```

returns 36 days.

FRCashFlowVsTotalDebt

Returns the ratio between cash flow and total debt.

FRCashFlowVsTotalDebt determines the ratio of cash flow to debt by dividing cash flow by total debt.

FRCashFlowVsTotalDebt (n1, n2)

n1 – numeric value indicating total cash flow.
n2 – numeric value indicating total debt.

```
FRCashFlowVsTotalDebt(100000,10000)
```

returns a cash flow–to–debt ratio of 10.

FRCurrentRatio

Returns the ratio between current assets and current liabilities.

FRCurrentRatio determines the ratio of current assets to current liabilities by dividing assets by liabilities.

FRCurrentRatio (n1, n2)

n1 – numeric value indicating total current assets.
n2 – numeric value indicating total current liabilities.

```
FRCurrentRatio(100000,10000)
```

returns an asset-to-liability ratio of 10.

FRDebtEquityRatio

Returns the ratio between total liabilities and total equity.

FRDebtEquityRatio determines the ratio of total liabilities to total equity by dividing liabilities by equity.

FRDebtEquityRatio (n1, n2)

n1 – numeric value indicating total liabilities.
n2 – numeric value indicating total equity.

```
FRDebtEquityRatio(10000,100000)
```

returns a liability-to-equity ratio of .1.

FRDividendYield

Returns the ratio between dividend and market price.

FRDividendYield determines the ratio of dividend to market price by dividing dividend by market price.

FRDividendYield (n1, n2)

n1 – numeric value indicating dividend amount.
n2 – numeric value indicating share market price.

```
FRDividendYield(10,50)
```

returns a dividend–to–market price ratio of .2.

FREarningsPerCommonShare

Returns the ratio of net profit distributable to shareholders to the number of shares.

FREarningsPerCommonShare determines the ratio of net profit to number of shares by subtracting the preferred dividend amount from net profit, and dividing the result by the number of common shares.

FREarningsPerCommonShare (n1, n2, n3)

n1 – numeric value indicating net profit.
n2 – numeric value indicating dividend paid to preferred shareholders.
n3 – numeric value indicating number of issued common shares.

```
FREarningsPerCommonShare (100000, 10, 10000)
```

returns a profit-to-share ratio of 10.

FREquityVsTotalAssets

Returns the ratio between total equity and total assets.

FREquityVsTotalAssets determines the ratio of total equity to total assets by dividing equity by assets.

FREquityVsTotalAssets (n1, n2)

n1 – numeric value indicating total equity.
n2 – numeric value indicating total assets.

```
FREquityVsTotalAssets(100000,500000)
```

returns a equity-to-assets ratio of .2.

FRGrossProfitMargin

Returns the ratio between gross profit and sales.

FRGrossProfitMargin determines the ratio of gross profit to sales by dividing gross profit by sales.

FRGrossProfitMargin (n1, n2)

n1 – numeric value indicating gross profit.
n2 – numeric value indicating sales.

```
FRGrossProfitMargin(10000,100000)
```

returns a profit-to-sales ratio of .1.

FRInterestCoverage

Returns the ratio between cash flow and interest expense.

FRInterestCoverage determines the ratio of cash flow to interest expense by dividing cash flow by interest expense.

FRInterestCoverage (n1, n2)

n1 – numeric value indicating cash flow.
n2 – numeric value indicating interest expense.

```
FRInterestCoverage(100000,2500)
```

returns a cash flow–to–interest expense ratio of 25.

FRInventoryTurnover

Returns the average turnover of inventory in numeric days.

FRInventoryTurnover calculates the average "turnover" of inventory by dividing inventory by sales, and multiplying the result by the number of days in the year.

FRInventoryTurnover (n1, n2, n3)

n1 – numeric value of total inventory.
n2 – numeric value of total sales.
n3 – numeric value indicating the number of days in a year (default is 360).

```
FRInventoryTurnover(10000,100000,360)
```

returns 36 days.

FRNetProfitMargin

Returns the ratio between net profit and sales.

FRNetProfitMargin determines the ratio of net profit to sales by dividing net profit by sales.

FRNetProfitMargin (n1, n2)

n1 – numeric value indicating net profit.
n2 – numeric value indicating sales.

```
FRNetProfitMargin(10000,100000)
```

returns a profit-to-sales ratio of .1.

FROperatingProfitMargin

Returns the ratio between operating profit and sales.

FROperatingProfitMargin determines the ratio of operating profit to sales by dividing operating profit by sales.

FROperatingProfitMargin (n1, n2)

n1 – numeric value indicating operating profit.
n2 – numeric value indicating sales.

```
FROperatingProfitMargin(10000,100000)
```

returns a profit-to-sales ratio of .1.

FRPriceEarningsRatio

Returns the ratio between share market price and earnings per share.

FRPriceEarningsRatio determines the ratio of share market price to earnings by dividing share market price by earnings per share.

FRPriceEarningsRatio (n1, n2)

n1 – numeric value indicating share market price.
n2 – numeric value indicating earnings per share.

```
FRPriceEarningsRatio (10,100)
```

returns a price-to-earnings ratio of .1.

FRQuickRatio

Returns the ratio between current assets (less inventory) and current liabilities.

FRQuickRatio determines the ratio of current assets (less inventory) to current liabilities by subtracting inventory from assets and dividing the results by liabilities.

FRQuickRatio (n1, n2, n3)

n1 – numeric value indicating current assets.
n2 – numeric value indicating inventory.
n3 – numeric value indicating current liabilities.

```
FRQuickRatio (100000, 10000, 5000)
```

returns a quick ratio of 18.

FRReturnOnCommonEquity

Returns the ratio between net profit and common equity.

FRReturnOnCommonEquity determines the ratio of net profit to common equity by subtracting the dividend amount from net profit, and dividing the result by common equity.

FRReturnOnCommonEquity (n1, n2, n3)

n1 – numeric value indicating net profit.
n2 – numeric value indicating the total preferred dividend amount.
n3 – numeric value indicating common equity.

```
FRReturnOnCommonEquity (10000, 15, 100000)
```

returns a return on common equity ratio of .1.

FRReturnOnEquity

Returns the ratio between net profit and total equity.

FRReturnOnEquity determines the ratio of net profit to total equity by dividing net profit by equity.

FRReturnOnEquity (n1, n2)

n1 – numeric value indicating net profit.
n2 – numeric value indicating total equity.

```
FRReturnOnEquity (10000,100000)
```

returns a profit-to-equity ratio of .1.

FRReturnOnInvestedCapital

Returns the ratio between net profit and invested capital.

FRReturnOnInvestedCapital determines the ratio between net profit and invested capital by dividing net profit by the sum of bank debt and equity.

FRReturnOnInvestedCapital (n1, n2, n3)

n1 – a numeric value indicating net profit.
n2 – a numeric value indicating total bank debt.
n3 – a numeric value indicating total equity.

```
FRReturnOnInvestedCapital (10000,5000,100000)
```

returns a return on invested capital ratio of .1.

FRReturnOnNetFixedAssets

Returns the ratio between net profit and net fixed assets.

FRReturnOnNetFixedAssets determines the ratio of net profit to fixed assets by dividing net profit by fixed assets.

FRReturnOnNetFixedAssets (n1, n2)

n1 – numeric value indicating net profit.
n2 – numeric value indicating net fixed assets.

```
FRReturnOnNetFixedAssets (10000,100000)
```

returns a profit-to-assets ratio of .1.

FRReturnOnTotalAssets

Returns the ratio between net profit and total assets.

FRReturnOnTotalAssets determines the ratio of net profit to total assets by dividing net profit by total assets.

FRReturnOnTotalAssets (n1, n2)

n1 – numeric value indicating net profit.
n2 – numeric value indicating total assets.

```
FRReturnOnTotalAssets (10000,75000)
```

returns a profit-to-assets ratio of .13.

LooksLike

Returns a Boolean (true or false) value based on whether the "mask" is found in the source string.

LooksLike uses DOS-style "wildcards" (the asterisk and question mark) to determine if the source string contains the characters specified in the wildcard-based mask. This is helpful for checking for partial text matches.

LooksLike (s1, s2)

s1 – the source string to be searched.
s2 – the mask string to search against. The mask can include a question mark wildcard to indicate a single-character substitution and/or an asterisk to indicate a multiple-character substitution.

```
LooksLike("George Peck", "G?orge*")
```

returns True.

Now

Returns the current time from the system clock as a text string.

This function is "left over" from previous versions of Crystal Reports and is similar to the CurrentTime function. The only difference between Now and CurrentTime is the data type returned. CurrentTime returns a time data type, while Now returns a string data type.

Picture

Formats a string based on the supplied formatting "mask" string.

Picture reformats a string value, inserting literal characters supplied within the "mask" formatting string. Picture is helpful for adding characters to an existing string, such as parentheses and hyphens to string database phone numbers without such punctuation.

Picture (s1, s2)

s1 – source string to format.
s2 – string "mask" to base formatting on. Any occurrence of an "x" character in the mask will substitute a single character from the source string. Any character other than "x" will simply be added to the output string.

```
Picture("8007733472", "Phone: (xxx) xxx-xxxx")
```

returns "Phone: (800) 773-3472".

Soundex

Returns a four-character string indicating how the source string "sounds".

Soundex uses a character-based algorithm to assign a "sound" to the source string. Similar-sounding strings, such as "George" and "gorge", will return the same four-character Soundex result. You may use Soundex in If-Then-Else statements, or similar constructs, where you want to compare the sound of two or more strings.

Soundex (s)

s – a string value to perform the Soundex algorithm on.

```
Soundex("George")
```

returns the string "G620".

```
If Soundex("George") = Soundex("gorge") Then
   "Sound the same"
Else
   "Different sound"
```

returns "Sound the same".

NOTE *An involved description of the Soundex algorithm can be found in Crystal Reports online help.*

Functions: Alerts

This set of functions exposes Crystal Reports Alerts to formulas.

AlertMessage

Returns a string containing the message portion of an alert.

AlertMessage returns the message portion of an alert when a particular report record has triggered the alert. If the record has not triggered the alert, AlertMessage returns an empty string.

AlertMessage (s)

s – a string value indicating the name of the alert.

```
AlertMessage("Order Over 5K")
```

returns "Orders over $5,000 exist" (the message for the Order Over 5K alert) if the report record has triggered the Order Over 5K alert.

AlertNames

Shows the name of each alert included in the report.

The AlertNames category of the Function tree (within the Alerts category) will show the names of each alert that has been created on the report. These alert names can be supplied as arguments to other Alert functions (described later in this section). If no alerts have been created, this function category won't appear in the Function tree.

IsAlertEnabled

Returns a Boolean (true or false) value indicating whether or not the "Enabled" check box is checked for the specified alert.

IsAlertEnabled tests to see if an alert will "fire" if records meet its criteria. An alert can be enabled or disabled when the alert is edited.

IsAlertEnabled (s)

s – a string value indicating the name of the alert.

```
If IsAlertEnabled("Order Over 5K") Then
    "Over 5K orders will be tested for"
```

returns "Over 5K orders will be tested for" if the enabled check box is checked in the Order Over 5K alert dialog box.

IsAlertTriggered

Returns a Boolean (true or false) value indicating whether or not an alert is triggered.

IsAlertTriggered is used within a formula to see if a particular alert has "fired" for the current record. The formula can base the remainder of its logic on whether or not the record has triggered the alert.

IsAlertTriggered (s)

s – a string value indicating the name of the alert.

```
If IsAlertTriggered ("Order Over 5K") Then
    "Congratulations--Great Order!"
```

returns "Congratulations—Great Order!" if the Order Over 5K alert is triggered when the formula evaluates.

Functions: Arrays

This category of functions applies to arrays. An *array* is a collection of data items stored in a single "bucket," such as a single variable. If an array contains 15 items, it is said to have 15 *elements.* Arrays can contain any supported Crystal Reports data type, such as number, string, date-time, and so forth.

Average

Returns a number containing the average of the numeric elements in the array.

Average (a)

a – an array. The array must be numeric.

```
Average([1,5,10,20,10])
```

returns 9.2.

Count

Returns a number containing the number of elements in the array.

Count is helpful when performing looping logic through an array. By using Count, you can determine how many array elements to loop through.

Count (a)

a – an array. The array can be of any data type.

```
Count([1,5,10,20,10])
```

returns 5.

NOTE *Count is functionally equivalent to UBound when dealing with arrays.*

DistinctCount

Returns a number containing the number of unique entries in the array.

DistinctCount is different from Count in that a duplicated entry in the array (for example, a second occurrence of the same element) will not increment the count when using DistinctCount, whereas it will with Count.

DistinctCount (a)

a – an array. The array can be of any data type.

```
DistinctCount(["George","Paul","George","John","Ringo","John"])
```

returns 4.

MakeArray

Creates an array.

MakeArray will create an array of values. It can be used to assign an array to a variable or pass a "literal" array to another array-based function. Supplying array elements within square brackets is the equivalent of MakeArray.

MakeArray (v1, v2...)

[v1, v2...]

v1 and v2 – literal values, separated by commas, that become array elements. The literal values can be any Crystal Reports data type, as long as they are all of the same data type. An array can contain, at maximum, 1,000 elements.

```
StringVar Array Beatles :=
    MakeArray("George","Paul","John","Ringo");
Beatles[3]
```

returns "John".

```
StringVar Array Beatles :=
    ["George","Paul","John","Ringo"];
Beatles[3]
```

returns "John".

NOTE *Basic syntax contains the Array function in addition to MakeArray. Array and MakeArray are functionally equivalent in Basic syntax.*

Maximum

Returns the last entry in an array.

Maximum will return the "last" entry in an array. What constitutes the "last" entry is based on the type of data in the array. Maximum will return the highest number in a numeric array, the last string value alphabetically in a string array, and so forth.

Maximum (a)

a – an array. The array can be of any data type.

```
Maximum([#1/1/1990#,#12/15/85#,#5/13/03#,#7/4/2000#])
```

will return 5/13/2003 12:00:00 A.M. as a date-time value.

Minimum

Returns the first entry in an array.

Minimum will return the "first" entry in an array. What constitutes the "first" entry is based on the type of data in the array. Minimum will return the lowest number in a numeric array, the first string value alphabetically in a string array, and so forth.

Minimum (a)

a – an array. The array can be of any data type.

```
Minimum(["Paul","George","John","Ringo"])
```

will return "George".

PopulationStdDev

Returns a number containing the population standard deviation of an array.

PopulationStdDev (a)

a – an array. The array must be numeric.

```
PopulationStdDev([1,10,15,20,25])
```

returns 8.28 (rounded to two decimal places).

PopulationVariance

Returns a number containing the population variance of an array.

PopulationVariance (a)

a – an array. The array must be numeric.

```
PopulationVariance([1,10,15,20,25])
```

returns 68.56 (rounded to two decimal places).

StdDev

Returns a number containing the standard deviation of an array.

StdDev (a)

a – an array. The array must be numeric.

```
StdDev([1,10,15,20,25])
```

returns 9.26 (rounded to two decimal places).

Sum

Returns a number containing the sum of array elements.

Sum (a)

a – an array. The array must be numeric.

```
Sum([1,10,15,20,25])
```

returns 71.

UBound

Returns the number of elements in the array.

UBound is helpful when performing looping logic through an array. By using UBound, you can determine how many array elements to loop through.

UBound (a)

a – an array. The array can be of any data type.

```
UBound([1,5,10,20,10])
```

returns 5.

NOTE *UBound is functionally equivalent to Count when dealing with arrays.*

Variance

Returns a number containing the variance of an array.

Variance (a)

a – an array. The array must be numeric.

```
Variance([1,10,15,20,25])
```

returns 85.70 (rounded to two decimal places).

Functions: Conditional Formatting

This category of functions appears in the Function tree only when you display the Format Formula Editor by clicking a Conditional Formatting Formula button from another Crystal Reports dialog box. Some of these functions will appear only in the proper "context"—for example, the Color or RGB functions will be available only when you are setting a color property conditionally.

Color

Returns a specific red-green-blue color combination.

Color allows you to specify intricate color values for font color, background color, and so forth. Rather than choosing from the limited set of default colors (such as crRed, crAqua, and so forth), you may specify any combination of red-green-blue colors with this function.

Color (n1, n2, n3)

n1 – a numeric value specifying the amount of red. The value must be between 0 and 255.
n2 – a numeric value specifying the amount of green. The value must be between 0 and 255.
n3 – a numeric value specifying the amount of blue. The value must be between 0 and 255.

```
If {Orders.Order Amount} > 5000 Then crRed Else Color(10,100,244)
```

will set the font color to a light blue shade (consisting of a red value of 10, green value of 100, and blue value of 244) if the order amount is not over 5,000.

NOTE *The RGB function is functionally equivalent to Color. RGB also accepts three arguments and may be used interchangeably with Color.*

CurrentFieldValue

Returns the contents of the field being formatted.

CurrentFieldValue will return the value contained in the field being formatted. The data type can vary, depending on the field being formatted. CurrentFieldValue is particularly valuable when conditionally formatting Cross-Tab or OLAP grid cells, as the cells themselves cannot be directly referred to in a formatting formula.

CurrentFieldValue

```
If CurrentFieldValue > 5000 Then crRed Else crBlack
```

will set the color property to red if the value of the object being formatted is greater than 5,000.

DefaultAttribute

Returns the setting of the formatting property from the originating dialog box.

DefaultAttribute returns the value set in the calling dialog box. For example, if you set the "absolute" font color to aqua but then click the Conditional Formatting Formula button, you will be able to recall the aqua value in the formatting formula by using DefaultAttribute.

DefaultAttribute

```
If {Orders.Order Amount} > 5000 Then crRed Else DefaultAttribute
```

will set the font color to that selected in the absolute font color property of the Format Editor if the order amount is not greater than 5,000.

GridRowColumnValue

Returns the value of the row or column name in a cross-tab or OLAP grid object.

GridRowColumnValue will return the contents of a "title" cell of a row or column in a cross-tab or OLAP grid object. By using this function, you may determine which row or column is currently formatting and change formatting logic accordingly.

GridRowColumnValue (s)

s – string value indicating row or column name to query.

```
If GridRowColumnValue ("Orders.Employee ID") = 3 Then
    crRed
Else
    crBlack
```

will determine if the Employee ID row in a cross-tab contain employee number 3. If so, the cell will be formatted in a red color.

Row Or Column Names

Names of the rows or columns in a cross-tab or OLAP grid object.

This category of the Function tree will appear only when you are conditionally formatting a cross-tab or OLAP grid object. The category will contain the names given to rows or columns in the cross-tab or OLAP grid. You may use these values with the GridRowColumnValue function to determine which row or column is currently formatting.

Functions: Date and Time

This set of functions deal with date and time manipulation, including extracting the current date and time from your computer's internal clock, performing date-time math, converting date-time functions to other data types, and so forth.

CurrentDate

Returns a date value with the current date from your computer's internal clock.

CurrentDate

```
CurrentDate
```

returns 5/30/2003 if the current date is May 30, 2003.

CurrentDateTime

Returns a date-time value with the current date and time from your computer's internal clock.

CurrentDateTime

```
CurrentDateTime
```

returns 5/30/2003 2:48:32PM if the current date is May 30, 2003 and the current time is 2:48 P.M.

CurrentTime

Returns a time value with the current time from your computer's internal clock.

CurrentTime

```
CurrentTime
```

returns 2:48:32PM if the current time is 2:48 in the afternoon.

Date

Returns a date value when supplied with various other data types.

Date is functionally equivalent to DateValue. See **DateValue** for details.

DateAdd

Returns a date-time value in the past or future, depending on supplied arguments.

DateAdd is handy for calculating dates in the past or future, based on a certain number of days, weeks, years, and so forth. DateAdd automatically handles leap years, months with less than 31 days, and so forth.

DateAdd (s, n, dt)

s – string value indicating interval type (days, weeks, and so forth). See **Interval Types** later in this section for recognized values.
n – a numeric value indicating number of intervals to add to the source date-time. This can be a negative or positive number.
dt – a date or date-time value indicating the source date-time to add intervals to.

```
DateAdd("d",5,CurrentDate)
```

returns 6/4/2003 12:00:00AM if the current date is May 30, 2003.

```
DateAdd("m",-2,#1/1/2003 10:00AM#)
```

returns 11/1/2002 10:00:00AM.

DateDiff

Returns a number indicating the number of "intervals" between two dates.

DateDiff is helpful in determining a specified amount of time that has elapsed between two date or date-time values. DateDiff can return the number of years, months, days, minutes, and so forth.

DateDiff (s, dt1, dt2)

s – string value indicating interval type (days, weeks, and so forth). See **Interval Types** later in this section for recognized values.
dt1 – a date, date-time, or time value indicating the start date-time to compare.
dt2 – a date, date-time, or time value indicating the end date-time to compare.

```
DateDiff ("h",#5/2/2003 10:00 AM#,#5/3/2003 2:00 PM#)
```

returns 28.

DateDiff (s, dt1, dt2, n)

s – string value indicating interval type (days, weeks, and so forth). See **Interval Types** later in this section for recognized values.
dt1 – a date, date-time, or time value indicating the start date-time to compare.
dt2 – a date, date-time, or time value indicating the end date-time to compare.
n – a numeric value or constant indicating the first day of the week. If not supplied, Sunday is assumed to be the first day of the week. See **First Day of Week Constants** later in this section for recognized constants or numbers for this argument.

```
DateDiff ("ww",#5/1/2003#,#6/1/2003#,crMonday)
```

returns 4.

DatePart

Returns a number indicating the individual "part" (month, hour, year, and so forth) of a date, date-time, or time value.

DatePart will return just the year, month, day, hour, minute, or second of a supplied date-time value. Other functions that work similarly to DatePart are Year, Month, Day, Hour, Minute, and Second.

DatePart (s, dt)

s – string value indicating interval type (day, week, hour, and so forth). See **Interval Types** later in this section for recognized values.
dt – a date, date-time, or time value to extract the interval from.

```
DatePart ("m",CurrentDate)
```

returns 5, if the current date is May 30, 2003.

DatePart (s, dt, n)

s – string value indicating interval type (day, week, hour, and so forth). See **Interval Types** later in this section for recognized values.
dt – a date, date-time, or time value to extract the interval from.
n – a numeric value or constant indicating the first day of the week. If not supplied, Sunday is assumed to be the first day of the week. See **First Day of Week Constants** later in this section for recognized constants or numbers for this argument.

```
DatePart ("w",#May 30, 2003#,crMonday)
```

returns 5, considering that May 30, 2003, is a Friday.

DatePart (s, dt, n1, n2)

s – string value indicating interval type (day, week, hour, and so forth). See **Interval Types** later in this section for recognized values.
dt – a date, date-time, or time value to extract the interval from.
n1 – a numeric value or constant indicating the first day of the week. If not supplied, Sunday is assumed to be the first day of the week. See **First Day of Week Constants** later in this section for recognized constants or numbers for this argument.
n2 – a numeric value or constant indicating the first week of year. If not supplied, the week containing January 1 is assumed to be the first week of the year. See **First Week of Year Constants** later in this section for recognized constants or numbers for this argument.

```
DatePart ("ww",#6/1/2003#,crSunday,crFirstJan1)
```

returns 23.

DateSerial

Returns a date value based on year, month, and day arguments.

DateSerial provides similar functionality to a specific usage of CDate, Date, and DateValue. DateSerial differs from these other functions in one significant way: the arguments supplied

do not have to fall into a strict "month between 1 and 12, day between 1 and 31" requirement. By supplying other values, such as negative numbers, calculations, and so forth, arguments will be evaluated "relatively."

DateSerial (n1, n2, n3)

n1 – a numeric value indicating the year.
n2 – a numeric value indicating the month.
n3 – a numeric value indicating the day.

```
DateSerial(2003, 6, 1)
```

returns 6/1/2003.

```
DateSerial(2003, 6-10, 1-30)
```

returns 7/2/2002 (10 months before June—which equates to August 1, 2002, and 30 days before the first day of the month, which equates to July 2, 2002).

DateTime

Returns a date-time value based on one of several sets and types of arguments.

DateTime is functionally equivalent to CDateTime and DateTimeValue. See **DateTimeValue** for more information.

DateTimeValue

Returns a date-time value based on one of several sets and types of arguments.

DateTimeValue can accept a wide variety of arguments that will be converted to a date-time. DateTimeValue comes in handy when you need to convert other data types to date-time, such as database fields from older systems that stored dates as strings. DateTimeValue is functionally equivalent to DateTime and CDateTime.

DateTimeValue (d)

d – a date value that will be converted to a date-time value with midnight as the time.

```
DateTimeValue(CurrentDate)
```

returns 5/30/2003 12:00:00AM if today's date is May 30, 2003 (in essence, converting the CurrentDate date-only value to a date-time value).

DateTimeValue (d, t)

d – a date value that will become the date portion of the resulting date-time value.
t – a time value that will become the time portion of the resulting date-time value.

```
DateTimeValue(CurrentDate, Time(#10:30 am#))
```

returns 5/30/2003 10:30:00AM if today's date is May 30, 2003.

DateTimeValue (n)

n – a numeric value indicating the number of days since December 30, 1899.

```
DateTimeValue(35000)
```

returns 10/28/1995 12:00:00AM (35,000 days from December 30, 1899).

DateTimeValue (s)

s – a string value that will be evaluated for particular date and time equivalents.

```
DateTimeValue("January 1, 2003 15:50")
```

returns 1/1/2003 3:50:00PM.

DateTimeValue (n1, n2, n3)

n1 – a numeric value indicating the year.
n2 – a numeric value between 1 and 12 indicating the month.
n3 – a numeric value between 1 and 31 indicating the day of month.

```
DateTimeValue(2002,10,15)
```

returns 10/15/2002 12:00:00AM.

DateTimeValue (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the year.
n2 – a numeric value between 1 and 12 indicating the month.
n3 – a numeric value between 1 and 31 indicating the day of month.
n4 – a numeric value between 0 and 23 indicating the hour of day.
n5 – a numeric value between 0 and 59 indicating the minute.
n6 – a numeric value between 0 and 59 indicating the second.

```
DateTimeValue(2002,10,15,14,35,50)
```

returns 10/15/2002 2:35:50PM.

DateValue

Returns a date value based on one of several sets and types of arguments.

DateValue can accept a wide variety of arguments that will be converted to a date. DateValue comes in handy when you need to convert other data types to date, such as database fields from older systems that stored dates as strings. DateValue is functionally equivalent to Date and CDate.

DateValue (dt)

dt – a date-time value that you wish to strip the time portion away from.

```
DateValue(CurrentDateTime)
```

returns 5/30/2002 if the current date is May 30, 2003, regardless of what time of day.

DateValue (n)

n – a numeric value indicating the number of days since December 30, 1899.

```
DateValue(35000)
```

returns 10/28/1995 (35,000 days from December 30, 1899).

DateValue (s)

s – a string value that will be evaluated for particular date equivalents.

```
DateValue("January 1, 2003")
```

returns 1/1/2003.

DateValue (n1, n2, n3)

n1 – a numeric value indicating the year.
n2 – a numeric value between 1 and 12 indicating the month.
n3 – a numeric value between 1 and 31 indicating the day of month.

```
DateValue(2002,10,15)
```

returns 10/15/2002.

Day

Returns a numeric value indicating the day of the month.

Day (dt)

dt – a date or date-time value.

```
Day(CurrentDate)
```

returns 30 if the current date is May 30, 2003.

DayOfWeek

Returns a numeric value indicating the day of the week.

DayOfWeek returns the day of the week. By default, DayOfWeek assumes Sunday is equal to 1. However, you may provide an extra argument to DayOfWeek indicating a different starting day. DayOfWeek is functionally equivalent to WeekDay.

DayOfWeek (dt)

dt – a date or date-time value.

```
DayOfWeek(CurrentDate)
```

returns 6 if the current date falls on a Friday.

DayOfWeek (dt, n)

dt – a date or date-time value.
n – a numeric value or constant indicating the first day of the week. If not supplied, Sunday is assumed to be the first day of the week. See **First Day of Week Constants** later in this section for recognized constants or numbers for this argument.

```
DayOfWeek(CurrentDate, crMonday)
```

returns 5 if the current date falls on a Friday.

Hour

Returns a numeric value indicating the hour of the supplied time or date-time in military time.

Hour (dt)

dt – a date-time or time value.

```
Hour(CurrentTime)
```

returns 17 if the current time is in the 5 P.M. hour.

IsDate

Returns a Boolean (true of false) value indicating whether or not the supplied number or string can be converted to a date value.

IsDate is helpful in determining if a supplied string or numeric value can be converted to a date with the DateValue (or similar) function. Because these other conversion functions will fail with a run-time error if an invalid value is supplied to them, using IsDate with an If-Then-Else statement can prevent run-time errors.

IsDate (s)

s – a string value to be evaluated for conversion to a date.

IsDate (n)

n – a numeric value to be evaluated as the number of days since December 30, 1899.

```
If IsDate({FromMainframe.OrderDateAsString}) Then
    DateValue({FromMainframe.OrderDateAsString})
Else
    DateValue(0,0,0)
```

returns a true order date if it can be converted, or 0/0/0 if it can't.

IsDateTime

Returns a Boolean (true of false) value indicating whether or not the supplied number or string can be converted to a date-time value.

IsDateTime is helpful in determining if a supplied string or numeric value can be converted to a date-time with the DateTimeValue (or similar) function. Because these other conversion

functions will fail with a run-time error if an invalid value is supplied to them, using IsDateTime with an If-Then-Else statement can prevent run-time errors.

IsDateTime (s)
s – a string value to be evaluated for conversion to a date-time.

IsDateTime (n)
n – a numeric value to be evaluated as the number of days since December 30, 1899.

```
If IsDateTime({FromMainframe.IncidentDateTimeAsString}) Then
    DateTimeValue({FromMainframe.IncidentDateTimeAsString})
Else
    DateTimeValue(0,0,0,0,0,0)
```

returns a true incident date-time if it can be converted, or 0/0/0 at midnight if it can't.

IsTime
Returns a Boolean (true of false) value indicating whether or not the supplied number or string can be converted to a time value.

IsTime is helpful in determining if a supplied string or numeric value can be converted to a time with the TimeValue (or similar) function. Because these other conversion functions will fail with a run-time error if an invalid value is supplied to them, using IsTime with an If-Then-Else statement can prevent run-time errors.

IsTime (s)
s – a string value to be evaluated for conversion to a time.

IsTime (n)
n – a numeric value to be evaluated as the number of "24-hour units".

```
If IsTime({FromMainframe.IncidentTimeAsString}) Then
    TimeValue({FromMainframe.IncidentTimeAsString})
Else
    TimeValue(0,0,0)
```

returns a true incident time if it can be converted, or midnight if it can't.

Minute
Returns a numeric value indicating the minute of the supplied time or date-time.

Minute (dt)
dt – a date-time or time value.

```
Minute(CurrentTime)
```

returns 43 if the current time 5:43 P.M.

Month

Returns a numeric value indicating the month.

Month (dt)

dt – a date or date-time value.

```
Month(CurrentDate)
```

returns 5 if the current date is May 30, 2003.

MonthName

Returns a string value indicating the spelled-out month of the supplied numeric argument.

MonthName (n)

n – a numeric value between 1 and 12 indicating the number of the month.

MonthName (n, b)

n – a numeric value between 1 and 12 indicating the number of the month.
b – a Boolean (true or false) value, indicating whether to abbreviate the month.

```
MonthName (3, True)
```

returns "Mar".

Second

Returns a numeric value indicating the second of the supplied time or date-time.

Second (dt)

dt – a date-time or time value.

```
Second(CurrentTime)
```

returns 45 if the current time 5:43:45 P.M.

Time

Returns a time value when supplied with various other data types.

Time is functionally equivalent to TimeValue. See **TimeValue** for details.

Timer

Returns the number of seconds since midnight.

Timer may be useful as a "seed" for the Rnd (random number) function.

Timer

```
Timer
```

returns 64,917 at 6:01:57 P.M.

TimeSerial

Returns a time value based on hour, minute, and second arguments.

TimeSerial provides similar functionality to a specific usage of CTime, Time, and TimeValue. TimeSerial differs from these other functions in one significant way: the arguments supplied do not have to fall into a strict "hour between 0 and 24, minute between 0 and 59" requirement. By supplying other values, such as negative numbers, calculations, and so forth, arguments will be evaluated "relatively."

TimeSerial (n1, n2, n3)

n1 – a numeric value indicating the hour.
n2 – a numeric value indicating the minute.
n3 – a numeric value indicating the second.

```
TimeSerial(17,30,15)
```

returns 5:30:15PM.

```
TimeSerial(17,30-40,15+25)
```

returns 4:50:40PM.

TimeValue

Returns a time value based on one of several sets and types of arguments.

TimeValue can accept a wide variety of arguments that will be converted to a time. TimeValue comes in handy when you need to convert other data types to time, such as database fields from older systems that stored times as strings. TimeValue is functionally equivalent to Time and CTime.

TimeValue (dt)

dt – a date-time value that you wish to strip the time portion away from.

```
TimeValue(CurrentDateTime)
```

returns 6:23:19PM if the current time is 6:23 P.M., regardless of what day.

TimeValue (n)

n – a numeric value to be evaluated as the number of "24-hour units."

```
TimeValue(.25)
```

returns 6:00:00AM.

TimeValue (s)

s – a string value that will be evaluated for particular time equivalents.

```
TimeValue("17:15")
```

returns 5:15:00PM.

TimeValue (n1, n2, n3)

n1 – a numeric value between 0 and 23 indicating the hour.
n2 – a numeric value between 0 and 59 indicating the minute.
n3 – a numeric value between 0 and 59 indicating the second.

```
TimeValue(14,12,0)
```

returns 2:12:00PM.

WeekdayName

Returns a string value containing a spelled-out name of the week.

WeekdayName will spell the name of the weekday ("Monday", etc.), based on a supplied numeric value.

WeekdayName (n)

n – numeric value between 1 and 7.

WeekdayName (n, b)

n – numeric value between 1 and 7.
b – a Boolean (true or false) value indicating whether to abbreviate the weekday name.

WeekdayName (n1, b, n2)

n1 – numeric value between 1 and 7.
b – Boolean (true or false) value indicating whether to abbreviate the weekday name.
n – numeric value or constant indicating the first day of the week. If not supplied, Sunday is assumed to be the first day of the week. See **First Day of Week Constants** later in this section for recognized constants or numbers for this argument.

```
WeekdayName (3, True, crMonday)
```

returns "Wed".

Year

Returns a numeric value indicating the year.

Year (dt)

dt – a date or date-time value.

```
Year(CurrentDate)
```

returns 2,003 if the current date is May 30, 2003.

Interval Types

The following interval types can be used with various date functions, such as DateAdd and DateDiff.

m	Month
d	Day (functionally equivalent to y)
yyyy	Year
y	Day of Year (functionally equivalent to d)
w	Weekday or Number of Weeks
ww	Week or Number of FirstDayOfWeeks
q	Quarter
h	Hour
m	Minute
s	Second

First Day of Week Constants

The following first-day-of-week values can be supplied to various functions that change output based on the first day of the week. Either the numeric value or the constant can be supplied.

crUseSystem	0 (default from Windows)
crSunday	1
crMonday	2
crTuesday	3
crWednesday	4
crThursday	5
crFriday	6
crSaturday	7

First Week of Year Constants

The following first-week-of-year values can be supplied to various functions that change output based on the first week of the year. Either the numeric value or the constant can be supplied.

crUseSystem	0 (default from Windows)
crFirstJan1	1
crFirstFourDays	2
crFirstFullWeek	3

Functions: Date Ranges

The following functions return date ranges based on the current date from the computer's clock. For point-in-time reporting purposes, you may change these ranges to be based on

something other than the current date by using Report | Set Print Date/Time. All these functions return range values and must be used with a range operator, such as In.

Aged0To30Days

Returns a range in the last 30 days.

Aged0To30Days

```
If {AR.Due Date} In Aged0To30Days Then {AR.Amount Due}
```

returns the AR Amount due if the due date is in the past 30 days. Otherwise, 0 is returned.

Aged31To60Days

Returns a range between 31 and 60 days in the past.

Aged31To60Days

```
If {AR.Due Date} In Aged31To60Days Then {AR.Amount Due}
```

returns the AR Amount due if the due date is within 31 to 60 days past. Otherwise, 0 is returned.

Aged61To90Days

Returns a range between 61 and 90 days in the past.

Aged61To90Days

```
If {AR.Due Date} In Aged61To90Days Then {AR.Amount Due}
```

returns the AR Amount due if the due date is within 61 to 90 days past. Otherwise, 0 is returned.

AllDatesFromToday

Returns a range from today into the future (including today).

AllDatesFromToday

```
If {Shipments.Ship Date} In AllDatesFromToday Then
   "Product still needs to ship"
Else
   "Product has already shipped"
```

returns the appropriate string, depending on whether the ship date is in the future or the past.

AllDatesFromTomorrow

Returns a range from tomorrow into the future (including tomorrow).

AllDatesFromTomorrow

```
If {Shipments.Ship Date} In AllDatesFromTomorrow Then
   "Product ships tomorrow or thereafter"
Else
   "Product has already shipped"
```

returns the appropriate string, depending on whether the ship date is tomorrow or later.

AllDatesToToday

Returns a range up to, and including, today.

AllDatesToToday

```
If {Shipments.Ship Date} In AllDatesToToday Then
   "Product has been shipped"
Else
   "Product ships later"
```

returns the appropriate string, depending on whether the product shipped today or earlier.

AllDatesToYesterday

Returns a range up to, and including, yesterday.

AllDatesToYesterday

```
If {Shipments.Ship Date} In AllDatesToYesterday Then
   "Product yesterday or before"
Else
   "Product ships today or later"
```

returns the appropriate string, depending on whether the product shipped yesterday or earlier.

Calendar1stHalf

Returns a range including January 1 to June 30 of the current year.

Calendar1stHalf

```
If {Sales.Sale Date} In Calendar1stHalf Then
    "Group 1: Jan 1 - Jun 30"
Else
    "Group 2: Jul 1 - Dec 31"
```

returns one of two appropriate values, depending on whether the sale took place in the first or second half of the year.

Calendar2ndHalf

Returns a range including July 1 to December 31 of the current year.

Calendar2ndHalf

```
If {Sales.Sale Date} In Calendar2ndHalf Then
    "Group 1: Jul 1 – Dec 31"
Else
    "Group 2: Jan 1 – Jun 30"
```

returns one of two appropriate values, depending on whether the sale took place in the first or second half of the year.

Calendar1stQtr

Returns a range including January 1 to March 31 of the current year.

Calendar1stQtr

```
CurrencyVar Q1Total;
If {Sales.Sale Date} In Calendar1stQtr Then
    Q1Total := Q1Total + {Sales.Amount}
```

increments a variable with a sales amount only if the sale took place in the first calendar quarter.

Calendar2ndQtr

Returns a range including April 1 to June 30 of the current year.

Calendar2ndQtr

```
CurrencyVar Q2Total;
If {Sales.Sale Date} In Calendar2ndQtr Then
    Q2Total := Q2Total + {Sales.Amount}
```

increments a variable with a sales amount only if the sale took place in the second calendar quarter.

Calendar3rdQtr

Returns a range including July 1 to September 30 of the current year.

Calendar3rdQtr

```
CurrencyVar Q3Total;
If {Sales.Sale Date} In Calendar3rdQtr Then
    Q3Total := Q3Total + {Sales.Amount}
```

increments a variable with a sales amount only if the sale took place in the third calendar quarter.

Calendar4thQtr

Returns a range including October 1 to December 31 of the current year.

Calendar4thQtr

```
CurrencyVar Q4Total;
If {Sales.Sale Date} In Calendar4thQtr Then
    Q4Total := Q4Total + {Sales.Amount}
```

increments a variable with a sales amount only if the sale took place in the fourth calendar quarter.

Last4WeeksToSun

Returns a range including the four weeks prior to, and including, the most recent Sunday.

Last4WeeksToSun, unlike some other date and date range functions, considers a week to start on Monday and end on Sunday.

Last4WeeksToSun

```
If {AR.Invoice Date} In Last4WeeksToSun Then
   "Current"
Else
   "Past Due"
```

returns a string indicating the status of an account receivable based on the last four weeks.

Last7Days

Returns a range consisting of the last seven days, including today.

Last7Days

```
If {Sales.Sale Date} In Last7Days Then
   "Eligible for Price Protection"
Else
   "Not Eligible for Price Protection"
```

returns a string indicating if a sale is eligible for price protection because it took place within the last seven days.

LastFullMonth

Returns a range including the first day of last month to the last day of last month.

LastFullMonth

```
If {Sales.Sale Date} In LastFullMonth Then
   "Eligible for Price Protection"
Else
   "Not Eligible for Price Protection"
```

returns a string indicating if a sale is eligible for price protection because it took place within the last month.

LastFullWeek

Returns a range including Sunday through Saturday of the previous week.

LastFullWeek

```
If {Sales.Sale Date} In LastFullWeek Then
   "Eligible for Price Protection"
Else
   "Not Eligible for Price Protection"
```

returns a string indicating if a sale is eligible for price protection because it took place within the previous Sunday through Saturday.

LastYearMTD

Returns a range including the first day of this month last year through today's date last year.

LastYearMTD

```
CurrencyVar LastYearMTDSales;
If {Sales.Sale Date}In LastYearMTD Then
    LastYearMTDSales := LastYearMTDSales + {Sales.Amount}
```

accumulates a sales amount in a variable if the sale took place this month-to-date last year.

LastYearYTD

Returns a range including January 1 last year through today's date last year.

LastYearYTD

```
CurrencyVar LastYearYTDSales;
If {Sales.Sale Date}In LastYearYTD Then
    LastYearYTDSales := LastYearYTDSales + {Sales.Amount}
```

accumulates a sales amount in a variable if the sale took place this year-to-date last year.

MonthToDate

Returns a range including the first day of this month through today.

MonthToDate

```
CurrencyVar MTDSales;
If {Sales.Sale Date}In MonthToDate Then
    MTDSales := MTDSales + {Sales.Amount}
```

accumulates a sales amount in a variable if the sale took place from the first of this month through today.

Next30Days

Returns a range in the next 30 days.

Next30Days

```
If {Shipments.Ship Date} In Next30Days Then
"High Priority – shipping within the month"
```

returns the string value if the shipment date is in the next 30 days. Otherwise, an empty string is returned.

Next31To60Days

Returns a range between 31 and 60 days in the future.

Next31To60Days

```
If {Shipments.Ship Date} In Next31To60Days Then
"Medium Priority – shipping 2 months from now"
```

returns the string value if the shipment date is between 31 and 60 days in the future. Otherwise, an empty string is returned.

Next61To90Days

Returns a range between 61 and 90 days in the future.

Next61To90Days

```
If {Shipments.Ship Date} In Next61To90Days Then
"Low Priority – shipping 3 months from now"
```

returns the string value if the shipment date is between 61 and 90 days in the future. Otherwise, an empty string is returned.

Next91To365Days

Returns a range between 91 and 365 days in the future.

Next91To365Days

```
If {Shipments.Ship Date} In Next91To365Days Then
"No Concern – shipping over 3 months from now"
```

returns the string value if the shipment date is between 91 and 365 days in the future. Otherwise, an empty string is returned.

Over90Days

Returns a range from 90 days in the past and beyond.

Over90Days

```
If {AR.Due Date} In Over90Days Then {AR.Amount Due}
```

returns the AR Amount due if the due date is over 90 days past the current date. Otherwise, 0 is returned.

WeekToDateFromSun

Returns a range starting the previous Sunday through today.

WeekToDateFromSun

```
If {Sales.Sale Date} In WeekToDateFromSun Then
   "Eligible for Price Protection"
Else
   "Not Eligible for Price Protection"
```

returns a string indicating if a sale is eligible for price protection because it took place between this past Sunday and today.

YearToDate

Returns a range including January 1 through today's date.

YearToDate

```
CurrencyVar ThisYearYTDSales;
If {Sales.Sale Date}In YearToDate Then
    ThisYearYTDSales := ThisYearYTDSales + {Sales.Amount}
```

accumulates a sales amount in a variable if the sale took place this year-to-date.

Functions: Document Properties

This category of functions duplicates many of the Special Fields available from the Field Explorer. By making these values available to formulas, you may base formulas on various properties of the report, such as the page number that's currently printing, whether a group header is being repeated, and so forth.

CurrentCEUserID

Returns a numeric value indicating the internal ID maintained by Crystal Enterprise for the currently logged-on user.

CECurrentUserID helps identify who is viewing the report after it has been published to Crystal Enterprise. While you may find some value in the numeric value this function returns, you'll probably find more value in the actual user name returned by the CurrentCEUserName function. This function returns a numeric zero if the report is not being displayed from within Crystal Enterprise (if used in stand-alone Crystal Reports, for example).

CECurrrentUserID

```
CECurrentUserID
```

returns the number 12.00 if the currently logged on user's internal ID number is 12. The function returns 0.00 if the report is not being displayed in Crystal Enterprise.

CurrentCEUserName

Returns a string value indicating the user name for the currently logged-on user.

CECurrentUserName helps identify who is viewing the report after it has been published to Crystal Enterprise. You may use this function to change report logic according to who is viewing the report in Crystal Enterprise. This may be helpful for creating certain limitations (similar to row and column security) based on the user. This function returns an empty string if the report is not being displayed from within Crystal Enterprise (if used in stand-alone Crystal Reports, for example).

CECurrrentUserName

```
If CECurrentUserName = "" Then
   "Not logged in to Crystal Enterprise"
Else
   CECurrentUserName
```

returns the user name of the current Crystal Enterprise user. If the report is being viewed outside Crystal Enterprise (for example, in the stand-alone Crystal Reports designer), the formula returns "Not logged in to Crystal Enterprise".

DataDate

Returns a date value indicating the last date that data on the report was refreshed from the database.

DataDate will be the same date that appears to the left of the page navigation controls in the Preview tab. Click Report | Refresh Report Data (or use the appropriate toolbar button or keyboard shortcut) to refresh the database and update this value to the current date.

DataDate

```
"Data current as of " & ToText(DataDate)
```

returns the string "Data current as of 5/31/2003" if the report was refreshed on May 31, 2003.

DataTime

Returns a time value indicating the last time that data on the report was refreshed from the database.

DataTime will be the same time that appears to the left of the page navigation controls in the Preview tab. Click Report | Refresh Report Data (or use the appropriate toolbar button or keyboard shortcut) to refresh the database and update this value to the current time.

DataTime

```
"Data current as of " & ToText(DataTime)
```

returns the string "Data current as of 6:15:21PM" if the report was refreshed at 6:15 P.M.

FileAuthor

Returns a string value from the Author field in the File | Summary Info dialog box.

FileAuthor

```
"Report written by " & FileAuthor
```

returns "Report written by George Peck", if "George Peck" has been placed in the Author field in the Summary Info dialog box.

FileCreationDate

Returns a date value indicating the date that the report file was first created.
This value will not change if the report is later opened, modified, and resaved.

FileCreationDate

```
"Report first created " & ToText(FileCreationDate)
```

will return "Report first created 5/30/2003" if the report was initially created on May 30, 2003.

FileName

Returns a string value containing the file path and filename of the report stored on disk.

FileName

```
"Report location: " & FileName
```

returns "Report location: C:\Reports\Sales Analysis.rpt" if the report is stored as C:\Reports\Sales Analysis.rpt.

ModificationDate

Returns a date value indicating the date that the report file was last modified.
This value will change any time the report is opened, modified, and resaved.

ModificationDate

```
"Report last modified on " & ToText(ModificationDate)
```

will return "Report last modified on 5/31/2003" if the report was changed on May 31, 2003.

ModificationTime

Returns a time value indicating the time that the report file was last modified.

This value will change any time the report is opened, modified, and resaved.

ModificationTime

```
"Report last modified at " & ToText(ModificationTime)
```

will return "Report last modified at 6:33:20PM" if the report was saved at 6:33 P.M.

PrintDate

Returns a date value indicating the date that the report is formatted or printed.

By default, this value is taken from your computer's system clock. You may change it to another value by choosing Report | Set Print Date And Time from the pull-down menus.

PrintDate

```
"Report printed on " & ToText(PrintDate)
```

will return "Report printed on 5/31/2003" if the report is printed or formatted on May 31, 2003.

PrintTime

Returns a time value indicating the time that the report is formatted or printed.

By default, this value is taken from your computer's system clock. You may change it to another value by choosing Report | Set Print Date And Time from the pull-down menus.

PrintTime

```
"Report printed at " & ToText(PrintTime)
```

will return "Report printed at 6:38:33PM" if the report is printed or formatted at 6:38 P.M.

ReportComments

Returns a string value from the Comments field in the File | Summary Info dialog box.

ReportComments

```
"Report Information: " & ReportComments
```

returns "Report Information: Ablaze Group, Version 1, May 2003", taking the text from the Comments field in the Summary Info dialog box.

ReportTitle

Returns a string value from the Title field in the File | Summary Info dialog box.

ReportTitle

```
"Report Title: " & ReportTitle
```

returns "Report Title: Sales Analysis", taking the text from the Title field in the Summary Info dialog box.

Functions: Evaluation Time

This category of functions is used to force a formula to evaluate during one of Crystal Reports' three available report "passes."

By default, formulas evaluate during a certain pass based on their contents. For example, a formula that makes no reference to database fields, such as:

```
"Report printed on " & PrintDate
```

prints in the BeforeReadingRecords (or preread) pass by default.

A formula that makes reference to one or more database fields, such as:

```
{Orders.Order Amount} * 1.10
```

prints in the WhileReadingRecords (or first) pass by default.

And, any formula that makes reference to summary functions, report alerts, and various other "second-pass" functions, such as:

```
{Orders.Order Amount} %
Sum({Orders.Order Amount}, {Employee. Last Name})
```

prints in the WhilePrintingRecords (or second) pass by default.

While you cannot force a formula to an earlier report pass than its default, you can force it to a later pass. In particular, when using variables in formulas, you'll often need to force formulas that use the same variable to the same report pass to ensure consistent results.

BeforeReadingRecords

Forces a formula to evaluate in the "preread," BeforeReadingRecords, report pass.

BeforeReadingRecords

```
BeforeReadingRecords;
NumberVar Array RegionTotals := [0,0,0,0,0,0,0,0,0,0];
0
```

will create an array variable and fill it with zeros before the report reads records from the database.

EvaluateAfter

Forces a formula to evaluate after another formula has been evaluated.

EvaluateAfter is required when you place two formulas in the same report section that evaluate in the same report pass. In this case, it's not predictable which formula will evaluate

first. If both formulas, for example, refer to the same variable, you may get unintended results if one formula doesn't evaluate after the other formula.

EvaluateAfter (f)

f – the name of another report formula.

```
EvaluateAfter({@Show Count});
NumberVar BonusCount := 0
```

resets the BonusCount variable to zero after the @Show Count formula has already processed.

WhilePrintingRecords

Forces a formula to evaluate during the second, WhilePrintingRecords, pass.

WhilePrintingRecords

```
WhilePrintingRecords;
NumberVar GroupBonusCount := 0
```

resets the GroupBonusCount variable to zero during the WhilePrintingRecords pass (presumably because other WhilePrintingRecords formulas make use of GroupBonusCount).

WhileReadingRecords

Forces a formula to evaluate during the first, WhileReadingRecords, pass.

WhileReadingRecords

```
WhileReadingRecords;
StringVar GroupField
```

forces the contents of the GroupField variable to be displayed during the WhileReadingRecords pass. This may be helpful if you wish to create report groups based on this formula.

Functions: Financial

This category of functions supplies many standard financial calculations and algorithms. Many of these functions behave similarly to those available in Microsoft Excel.

ACCRINT

Returns a numeric value indicating total accrued interest for a security.

ACCRINT (d1, d2, d3, n1, n2, n3)

d1 – a date or date-time value indicating the issue date.
d2 – a date or date-time value indicating the date of first interest payment.
d3 – a date or date-time value indicating the date the security was purchased.
n1 – a numeric value indicating the annual interest rate.

n2 – a numeric value indicating the par value.
n3 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

ACCRINT (d1, d2, d3, n1, n2, n3, n4)

d1 – a date or date-time value indicating the issue date.
d2 – a date or date-time value indicating the date of first interest payment.
d3 – a date or date-time value indicating the date the security was purchased.
n1 – a numeric value indicating the annual interest rate.
n2 – a numeric value indicating the par value.
n3 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n4 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
ACCRINT(#1/1/1995#, #1/1/1996#, #6/1/1995#, .065, 100000, 4, 3)
```

returns 2,711.30.

ACCRINTM

Returns a numeric value indicating total accrued interest for a security that pays interest when it matures.

ACCRINTM (d1, d2, n1, n2)

d1 – a date or date-time value indicating the issue date.
d2 – a date or date-time value indicating the maturity date.
n1 – a numeric value indicating the annual interest rate.
n2 – a numeric value indicating the par value.

ACCRINTM (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the issue date.
d2 – a date or date-time value indicating the maturity date.
n1 – a numeric value indicating the annual interest rate.
n2 – a numeric value indicating the par value.
n3 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
ACCRINTM(#1/1/1995#, #1/1/2005#, .065, 100000, 3)
```

returns 65,053.42.

AmorDEGRC

Returns a numeric value indicating depreciation of an asset.

AmorDEGRC (n1, d1, d2, n2, n3, n4)

n1 – a numeric value indicating the asset's initial cost.
d1 – a date or date-time value indicating the asset's purchase date.

d2 – a date or date-time value indicating the end of the first period.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the period to calculate the depreciation for.
n4 – a numeric value indicating the depreciation rate.

AmorDEGRC (n1, d1, d2, n2, n3, n4, n5)

n1 – a numeric value indicating the asset's initial cost.
d1 – a date or date-time value indicating the asset's purchase date.
d2 – a date or date-time value indicating the end of the first period.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the period to calculate the depreciation for.
n4 – a numeric value indicating the depreciation rate.
n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
AmorDEGRC (35000,#5/1/2000#,#1/1/2001#,15000,1,.1)
```

returns 7,292.00.

AmorLINC

Returns a numeric value indication linear depreciation of an asset.

AmorLINC (n1, d1, d2, n2, n3, n4)

n1 – a numeric value indicating the asset's initial cost.
d1 – a date or date-time value indicating the asset's purchase date.
d2 – a date or date-time value indicating the end of the first period.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the period to calculate the depreciation for.
n4 – a numeric value indicating the depreciation rate.

AmorLINC (n1, d1, d2, n2, n3, n4, n5)

n1 – a numeric value indicating the asset's initial cost.
d1 – a date or date-time value indicating the asset's purchase date.
d2 – a date or date-time value indicating the end of the first period.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the period to calculate the depreciation for.
n4 – a numeric value indicating the depreciation rate.
n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
AmorLINC (35000,#5/1/2000#,#1/1/2001#,15000,1,.1)
```

returns 3,500.00.

CoupDayBS

Returns a numeric value indicating the days between the last coupon date and the settlement date.

CoupDayBS (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.
n – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.

CoupDayBS (d1, d2, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.
n1 – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n2 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
CoupDayBS (#1/1/2000#, #5/31/2004#, 2, 3)
```

returns 32.

CoupDays

Returns a numeric value indicating the number of days in the coupon period that includes settlement.

CoupDays (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.
n – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.

CoupDays (d1, d2, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.
n1 – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n2 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
CoupDays (#1/1/2000#, #5/31/2004#, 2, 3)
```

returns 182.5.

CoupDaysNC

Returns a numeric value indicating the number of days between the settlement date and the next coupon date.

CoupDaysNC (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.

n – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.

CoupDaysNC (d1, d2, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.
n1 – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n2 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
CoupDaysNC (#1/1/2000#, #5/31/2004#, 2, 3)
```

returns 151.

CoupNCD

Returns a date value indicating the next coupon date after settlement.

CoupNCD (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.
n – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.

```
CoupNCD (#1/1/2000#, #5/31/2004#, 2)
```

returns 5/31/2000.

CoupNum

Returns a numeric value indicating the number of coupon periods between settlement and maturity.

CoupNum (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the maturity date.
n – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.

```
CoupNum (#1/1/2000#, #5/31/2004#, 2)
```

returns 9.

CoupPCD

Returns a date value indicating the last coupon date before settlement.

CoupPCD (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.

d2 – a date or date-time value indicating the maturity date.
n – a numeric value indicating the number of payments (coupons) per year: 1 = annual, 2 = semiannual, 4 = quarterly.

```
CoupPCD (#1/1/2000#, #5/31/2004#, 2)
```

returns 11/30/1999.

CumIPMT

Returns a numeric value indicating cumulative interest paid on a loan.

CumIPMT (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the interest rate, which must be provided on a per-period basis.
n2 – a numeric value indicating the total number of payment periods, which must be provided on a per-period basis.
n3 – a numeric value indicating the loan's total or "present" value of the loan.
n4 – a numeric value indicating the first period to include in the calculation.
n5 – a numeric value indicating the last period to include in the calculation.
n6 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

```
CumIPMT (.09/12, 60, 45000, 1, 30, 1)
```

returns –7,640.28.

CumPrinc

Returns a numeric value indicating cumulative principal paid on a loan.

CumPrinc (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the interest rate, which must be provided on a per-period basis.
n2 – a numeric value indicating the total number of payment periods, which must be provided on a per-period basis.
n3 – a numeric value indicating the loan's total or "present" value of the loan.
n4 – a numeric value indicating the first period to include in the calculation.
n5 – a numeric value indicating the last period to include in the calculation.
n6 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

```
CumPrinc (.09/12, 60, 45000, 1, 30, 1)
```

returns –20,174.89.

Days360

Returns a numeric value indicating the number of days between two dates using a 30-day-per-month/360-day-per-year financial calendar.

Days360 (d1, d2)

d1 – a date or date-time value indicating start date.
d2 – a date or date-time value indicating end date.

Days360 (d1, d2, b)

d1 – a date or date-time value indicating start date.
d2 – a date or date-time value indicating end date.
b – a Boolean (true or false) value indicating the basis for the calculation: False indicates US (NASD) 30/360; True indicates European 30/360.

```
Days360(#1/1/2003#, #12/31/2003#)
```

returns 360.

DB

Returns a numeric value indicating depreciation of an asset using the declining balance method.

DB (n1, n2, n3, n4)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the period to calculate the depreciation for.

DB (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the period to calculate the depreciation for.
n5 – a numeric value indicating the number of months in the first year.

```
DB(35000,15000,7,5)
```

returns 2,458.71.

DDB

Returns a numeric value indicating depreciation of an asset using the double-declining balance method.

DDB (n1, n2, n3, n4)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the period to calculate the depreciation for.

DDB (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the period to calculate the depreciation for.
n5 – a numeric value indicating the factor at which the rate declines; 2 indicates double-declining.

```
DDB(35000,5000,7,5)
```

returns 2,603.08.

DISC

Returns a numeric value indicating the discount rate for a security.

DISC (d1, d2, n1, n2)

d1 – a date or date-time value indicating the date the security was purchased.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the security's value when purchased.
n2 – a numeric value indicating the security's value at maturity.

DISC (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the date the security was purchased.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the security's value when purchased.
n2 – a numeric value indicating the security's value at maturity.
n3 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
DISC(#1/1/2000#,#12/31/2005#,10000,14000)
```

returns .05.

DollarDE

Returns a number indicating the true fractional value of a financial fractional "representation."

This function is helpful for calculating actual decimal values of such terms as "Five and seven eighths."

DollarDE (n1, n2)

n1 – a numeric value representing the number to be converted (such as 5.7 for "five and seven").
n2 – a numeric value representing the base of n1 (such as 8 for "eighths").

```
DollarDE (5.7, 8)
```

returns 5.88 (5.875 rounded to two decimal places).

DollarFR

Returns a number indicating the financial fractional "representation" of a true fractional value.

This function is helpful for converting actual decimal values into such terms as "Five and seven eighths."

DollarFR (n1, n2)

n1 – a numeric value representing the true fractional number to be converted (such as 5.875).
n2 – a numeric value representing the base of to convert n1 to (such as 8 for "eighths").

```
DollarFR (5.875, 8)
```

returns 5.70 (indicating "five and seven eighths").

Duration

Returns a number indicating the duration of a bond (sometimes known as the "Macauley duration").

Duration (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the bond's purchase date.
d2 – a date or date-time value indicating the bond's maturity date.
n1 – a numeric value indicating the bond's interest rate.
n2 – a numeric value indicating the bond's yield.
n3 – a numeric value indicating the number of payments per year; 1 = annual, 2 = semiannual, 4 = quarterly.

Duration (d1, d2, n1, n2, n3, n4)

d1 – a date or date-time value indicating the bond's purchase date.
d2 – a date or date-time value indicating the bond's maturity date.
n1 – a numeric value indicating the bond's interest rate.
n2 – a numeric value indicating the bond's yield.
n3 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n4 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
Duration(#1/1/2000#,#1/1/2030#,.08,.09,1)
```

returns 11.37 years.

Effect

Returns a numeric value indicating the effective annual interest rate.

Effect (n1, n2)

n1 – a numeric value indicating the annual interest rate.

n2 – a numeric value indicating the number of compounding periods.

```
Effect(.1,12)
```

returns 0.10471 (when formatted to show five decimal places).

FV

Returns a numeric value indicating the future value of an annuity.

FV (n1, n2, n3)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the number of annuity payments.
n3 – a numeric value indicating the payment amount.

FV (n1, n2, n3, n4)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the number of annuity payments.
n3 – a numeric value indicating the payment amount.
n4 – a numeric value indicating the present value (the total amount that a series of future payments is worth now).

FV (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the number of annuity payments.
n3 – a numeric value indicating the payment amount (a negative amount indicating paying money "out").
n4 – a numeric value indicating the present value (the total amount that a series of future payments is worth now).
n5 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

```
FV (.04/12, 60, -500)
```

returns 33,149.49.

FVSchedule

Returns a number indicating the future value of an annuity with several compounded interest rates.

FVSchedule (n, a)

n – a numeric value indicating the present value of the annuity.
a – a numeric array containing one or more interest rates.

```
FVSchedule (10000,[.04,.0475,.05])
```

returns 11,438.70.

IntRate

Returns a numeric value indicating the interest rate for a security.

IntRate (d1, d2, n1, n2)

d1 – a date or date-time value indicating the security purchase date.
d2 – a date or date-time value indicating the security maturity date.
n1 – a numeric value indicating the original security value.
n2 – a numeric value indicating the value at maturity.

IntRate (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the security purchase date.
d2 – a date or date-time value indicating the security maturity date.
n1 – a numeric value indicating the original security value.
n2 – a numeric value indicating the value at maturity.
n3 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
IntRate(#1/1/2000#,#1/1/2005#,10000,13000)
```

returns .06.

IPmt

Returns a numeric value indicating an interest payment based on fixed payments and interest rate.

IPmt (n1, n2, n3, n4)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the payment period.
n3 – a numeric value indicating the total number of payments.
n4 – a numeric value indicating the present value.

IPmt (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the payment period.
n3 – a numeric value indicating the total number of payments.
n4 – a numeric value indicating the present value.
n5 – a numeric value indicating the future value after the last payment.

IPmt (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the payment period.
n3 – a numeric value indicating the total number of payments.
n4 – a numeric value indicating the present value.
n5 – a numeric value indicating the future value after the last payment.

n6 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

```
IPmt(.09/12,30,60,30000)
```

returns –128.76.

IRR

Returns a numeric value indicating the internal rate of return for a series of payments and receipts.

IRR (a)

a – a numeric array indicating the series of payments and receipts (there must be at least one negative and one positive number in the array). An IRR of 10 percent (.1) is assumed.

IRR (a, n)

a – a numeric array indicating the series of payments and receipts (there must be at least one negative and one positive number in the array).
n – a numeric value indicating the "guess" of IRR.

```
IRR([-1000,-500,2000])
```

returns .19.

ISPMT

Returns a numeric value indicating interest paid during a period (assuming equal installments).

ISPMT (n1, n2, n3, n4)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the period to calculate interest for.
n3 – a numeric value indicating the number of payment periods.
n4 – a numeric value indicating the present value.

```
ISPMT(.09/12,30,60,30000)
```

returns –112.50.

MDuration

Returns a number indicating the modified duration of a bond (sometimes known as the modified "Macauley duration").

MDuration (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the bond's purchase date.
d2 – a date or date-time value indicating the bond's maturity date.
n1 – a numeric value indicating the bond's interest rate.

n2 – a numeric value indicating the bond's yield.
n3 – a numeric value indicating the number of payments per year; 1 = annual, 2 = semiannual, 4 = quarterly.

MDuration (d1, d2, n1, n2, n3, n4)

d1 – a date or date-time value indicating the bond's purchase date.
d2 – a date or date-time value indicating the bond's maturity date.
n1 – a numeric value indicating the bond's interest rate.
n2 – a numeric value indicating the bond's yield.
n3 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n4 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
MDuration(#1/1/2000#,#1/1/2030#,.08,.09,1)
```

returns 10.43 years.

MIRR

Returns a numeric value indicating the modified internal rate of return for a series of payments and receipts.

MIRR (a, n1, n2)

a – a numeric array indicating the series of payments and receipts (there must be at least one negative and one positive number in the array).
n1 – a numeric value indicating interest rate cost.
n2 – a numeric value indicating interest rate return.

```
MIRR([-1000,-500,2000], .9, .6)
```

returns .26.

Nominal

Returns a numeric value indicating the nominal annual interest rate.

Nominal (n1, n2)

n1 – a numeric value indicating effective rate.
n2 – a numeric value indicating number of compounding periods.

```
Nominal(.08,12)
```

returns .07721 (when formatted with five decimal places).

NPer

Returns a numeric value indicating the number of payments, based on a fixed number of payments and interest rate.

NPer (n1, n2, n3)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period value).
n2 – a numeric value indicating the combined principal/interest payment.
n3 – a numeric value indicating the present value of the loan.

NPer (n1, n2, n3, n4)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period value).
n2 – a numeric value indicating the combined principal/interest payment.
n3 – a numeric value indicating the present value of the loan.
n4 – a numeric value indicating the future value of the loan after final payment.

NPer (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period value).
n2 – a numeric value indicating the combined principal/interest payment.
n3 – a numeric value indicating the present value of the loan.
n4 – a numeric value indicating the future value of the loan after final payment.
n5 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

```
NPer(.09/12,-750,40000)
```

returns 68.37 (indicating months).

NPV

Returns a numeric value indicating the net present value of an investment.

NPV (n, a)

n – a numeric value indicating the discount rate.
a – a numeric array indicating payments/receipts.

```
NPV(.05,[1000,2000,1500,1750])
```

returns 5,501.93.

OddFPrice

Returns a numeric value indicating the price of a security that pays interest with regularity, with the exception of a short or long (odd) first period.

OddFPrice (d1, d2, d3, d4, n1, n2, n3, n4)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.
d4 – a date or date-time value indicating the date of first interest payment.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the annual yield of the security.
n3 – a numeric value indicating the security redemption value (per $100 face value).

n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

OddFPrice (d1, d2, d3, d4, n1, n2, n3, n4, n5)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.
d4 – a date or date-time value indicating the date of first interest payment.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the annual yield of the security.
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
OddFPrice (#2/1/2000#,#1/1/2030#,#1/1/2000#,
   #1/1/2001#,.05,.055,75,1)
```

returns 87.69.

OddFYield

Returns a numeric value indicating the yield of a security that pays interest with regularity, with the exception of a short or long (odd) first period.

OddFYield (d1, d2, d3, d4, n1, n2, n3, n4)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.
d4 – a date or date-time value indicating the date of first interest payment.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the purchase price (per $100 face value).
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

OddFYield (d1, d2, d3, d4, n1, n2, n3, n4, n5)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.
d4 – a date or date-time value indicating the date of first interest payment.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the purchase price (per $100 face value).
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
OddFPrice (#2/1/2000#,#1/1/2030#,#1/1/2000#,
   #1/1/2001#,.05,13.23,75,1)
```

returns .0549 (formatted with four decimal places).

OddLPrice

Returns a numeric value indicating the price of a security that pays interest with regularity, with the exception of a short or long (odd) last period.

OddLPrice (d1, d2, d3, n1, n2, n3, n4)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the security's last coupon date.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the annual yield of the security.
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

OddLPrice (d1, d2, d3, n1, n2, n3, n4, n5)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the security's last coupon date.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the annual yield of the security.
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
OddLPrice(#2/1/2000#,#1/1/2030#,#1/1/2000#,.05,.055,75,1)
```

returns 84.64.

OddLYield

Returns a numeric value indicating the yield of a security that pays interest with regularity, with the exception of a short or long (odd) last period.

OddLYield (d1, d2, d3, n1, n2, n3, n4)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the security's last coupon date.
n1 – a numeric value indicating the interest rate.

n2 – a numeric value indicating the purchase price (per $100 face value).
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

OddLYield (d1, d2, d3, n1, n2, n3, n4, n5)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the security's last coupon date.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the purchase price (per $100 face value).
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
OddLYield (#2/1/2000#,#1/1/2030#,#1/1/2000#,.05,84.64,75,1)
```

returns .055 (formatted with three decimal places).

Pmt

Returns a numeric value indicating the payment for an annuity with fixed payments and interest rate.

Pmt (n1, n2, n3)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the number of payment periods.
n3 – a numeric value indicating the principal/present value of the annuity.

Pmt (n1, n2, n3, n4)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the number of payment periods.
n3 – a numeric value indicating the principal/present value of the annuity.
n4 – a numeric value indicating the desired future value/balance after the final payment.

Pmt (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the number of payment periods.
n3 – a numeric value indicating the principal/present value of the annuity.
n4 – a numeric value indicating the desired future value/balance after the final payment.
n5 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

```
Pmt(.05/12,36,10000)
```

returns –299.71.

PPmt

Returns a numeric value indicating the principal payment for a specific period of an annuity with fixed payments and interest rate.

PPmt (n1, n2, n3, n4)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the specific period to calculate the principal for.
n3 – a numeric value indicating the number of payment periods.
n4 – a numeric value indicating the present value.

PPmt (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the specific period to calculate the principal for.
n3 – a numeric value indicating the number of payment periods.
n4 – a numeric value indicating the present value.
n5 – a numeric value indicating the desired future value/balance after the final payment.

PPmt (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the interest rate (must be adjusted to a per-period rate).
n2 – a numeric value indicating the specific period to calculate the principal for.
n3 – a numeric value indicating the number of payment periods.
n4 – a numeric value indicating the present value.
n5 – a numeric value indicating the desired future value/balance after the final payment.
n6 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

```
PPmt(.09/12,24,60,35000)
```

returns –551.05.

Price

Returns a numeric value indicating the price of a security (per $100 face value) that pays periodic interest.

Price (d1, d2, n1, n2, n3, n4)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the annual yield of the security.
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

Price (d1, d2, n1, n2, n3, n4, n5)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the interest rate.

n2 – a numeric value indicating the annual yield of the security.
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.
n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
Price (#1/1/2000#,#1/1/2005#,.055,.05,50,1)
```

returns 62.99.

PriceDisc

Returns a numeric value indicating the discounted price of a security (per $100 face value).

PriceDisc (d1, d2, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the discount rate.
n2 – a numeric value indicating the security redemption value (per $100 face value).

PriceDisc (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the discount rate.
n2 – a numeric value indicating the security redemption value (per $100 face value).
n3 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
PriceDisc (#1/1/2000#,#1/1/2005#,.055,50)
```

returns 36.25.

PriceMat

Returns a numeric value indicating the price of a security (per $100 face value) that pays interest only at maturity.

PriceMat (d1, d2, d3, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the security's yield.

PriceMat (d1, d2, d3, n1, n2, n3)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.

Rate (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the number of annuity payments.
n2 – a numeric value indicating the payment amount.
n3 – a numeric value indicating the annuity's present value.
n4 – a numeric value indicating the future value/balance after final payment.
n5 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.

Rate (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the number of annuity payments.
n2 – a numeric value indicating the payment amount.
n3 – a numeric value indicating the annuity's present value.
n4 – a numeric value indicating the future value/balance after final payment.
n5 – a numeric value indicating the type of payment: 0 = end of payment period, 1 = beginning of payment period.
n6 – a numeric value indicating your "guess" of the return to use as a starting point for the calculation. If you omit this argument, 10 percent (.1) is assumed.

```
Rate(36, -750, 35000)
```

returns .0134 (when formatted to four decimal places).

Received

Returns a numeric value indicating the amount received at maturity for a fully invested security.

Received (d1, d2, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the investment amount.
n2 – a numeric value indicating the discount rate.

Received (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the investment amount.
n2 – a numeric value indicating the discount rate.
n3 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
Received(#1/1/2000#,#1/1/2005#,10000,.075)
```

returns 16,000.00.

SLN

Returns a numeric value indicating straight-line depreciation of an asset.

SLN (n1, n2, n3)

n1 – a numeric value indicating original cost of the asset.
n2 – a numeric value indicating the salvage of the asset at the end of its life.
n3 – a numeric value indicating the life of the asset.

```
SLN(35000,7500,7)
```

returns a per-year depreciation of 3,928.57.

SYD

Returns a numeric value indicating sum-of-years-digits depreciation of an asset.

SYD (n1, n2, n3, n4)

n1 – a numeric value indicating original cost of the asset.
n2 – a numeric value indicating the salvage of the asset at the end of its life.
n3 – a numeric value indicating the life of the asset.
n4 – a numeric value indicating the period to calculate depreciation for.

```
SYD(35000,7500,7,2)
```

returns a second-year depreciation of 5,892.86.

TBillEq

Returns a numeric value indicating the bond-equivalent yield for a Treasury bill.

TBillEq (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n – a numeric value indicating the discount rate.

```
TBillEq(#1/1/2000#,#12/31/2000#,.08)
```

returns .0864 (8.64%) when formatted with four decimal places.

TBillPrice

Returns a numeric value indicating the price for a Treasury bill (per $100 face value).

TBillPrice (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n – a numeric value indicating the discount rate.

```
TBillPrice(#1/1/2000#,#12/31/2000#,.08)
```

returns 91.89.

TBillYield

Returns a numeric value indicating the yield for a Treasury bill.

TBillYield (d1, d2, n)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n – a numeric value indicating the price per $100 face value.

```
TBillYield(#1/1/2000#,#12/31/2000#,91.89)
```

returns .087 (8.7%) when formatted with three decimal places.

VDB

Returns the variable declining balance for an asset.

VDB supports various depreciation options, such as double-declining balance or other method and partial depreciation periods.

VDB (n1, n2, n3, n4, n5)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the start period to calculate depreciation for.
n5 – a numeric value indicating the end period to calculate depreciation for.

VDB (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the start period to calculate depreciation for.
n5 – a numeric value indicating the end period to calculate depreciation for.
n6 – a numeric value indicating the factor at which the rate declines – 2 indicates double-declining.

VDB (n1, n2, n3, n4, n5, n6)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the start period to calculate depreciation for.
n5 – a numeric value indicating the end period to calculate depreciation for.
n6 – a Boolean (true/false) value indicating whether to switch from declining-balance to straight-line if declining-balance is less. True will always use declining-balance. False will switch to straight-line if declining-balance is less.

VDB (n1, n2, n3, n4, n5, n6, n7)

n1 – a numeric value indicating the asset's initial cost.
n2 – a numeric value indicating the asset's salvage value.
n3 – a numeric value indicating the useful life of the asset.
n4 – a numeric value indicating the start period to calculate depreciation for.
n5 – a numeric value indicating the end period to calculate depreciation for.
n6 – a numeric value indicating the factor at which the rate declines – 2 indicates double-declining.
n7 – a Boolean (true/false) value indicating whether to switch from declining-balance to straight-line if declining-balance is less. True will always use declining-balance. False will switch to straight-line if declining-balance is less.

```
VDB(50000,15000,7,2,5)
```

returns 10,510.20 depreciation for years two through five.

XIRR

Returns a numeric value indicating the internal rate of return for a series of payments and receipts.

XIRR differs from IRR in that it calculates internal rate of return for payments/receipts that aren't necessarily periodic.

XIRR (na, da)

na – a numeric array indicating the series of payments and receipts (there must be at least one negative and one positive number in the array).
da – an array of date or date-time values indicating the matching dates for the numeric array. There must be one entry in this array for each entry in the numeric array.

XIRR (na, da, n)

na – a numeric array indicating the series of payments and receipts (there must be at least one negative and one positive number in the array).
da – an array of date or date-time values indicating the matching dates for the numeric array. There must be one entry in this array for each entry in the numeric array.
n – a numeric value indicating the "guess" of IRR.

```
XIRR([-1000,-500,2000],[#1/1/2000#,#5/1/2000#,#6/30/2000#])
```

returns 1.08.

XNPV

Returns a numeric value indicating the net present value of an investment.

XNPV differs from NPV in that it calculates net present value for payments/receipts that aren't necessarily periodic.

XNPV (n, na, da)

n – a numeric value indicating the discount rate.

na – a numeric array indicating payments/receipts.
da – an array of date or date-time values that correspond to the numeric array. There must be one entry in the date array for each entry in the numeric array.

```
XNPV(.05,[1000,2000,1500,1750],
[#1/1/2000#,#5/1/2000#,#6/30/2000#,#12/31/2000#])
```

returns 6,098.72.

YearFrac

Returns a numeric value indicating what fraction of the year is represented by two dates.

YearFrac (d1, d2)

d1 – a date or date-time value indicating the first date in the year.
d2 – a date or date-time value indicating the second date in the year.

YearFrac (d1, d2, n)

d1 – a date or date-time value indicating the first date in the year.
d2 – a date or date-time value indicating the second date in the year.
n – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
YearFrac(#1/1/2003#,#3/31/2003#)
```

returns .25.

Yield

Returns a numeric value indicating the yield of a security that pays periodic interest.

Yield (d1, d2, n1, n2, n3, n4)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the purchase price of the security (per $100 face value).
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

Yield (d1, d2, n1, n2, n3, n4, n5)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating the purchase price of the security (per $100 face value).
n3 – a numeric value indicating the security redemption value (per $100 face value).
n4 – a numeric value indicating the number of payments per year: 1 = annual, 2 = semiannual, 4 = quarterly.

n5 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
Yield (#1/1/2000#,#1/1/2005#,.055,62.99,50,1)
```

returns .087 (formatted with three decimal places).

YieldDisc

Returns a numeric value indicating the yield of a discounted security.

YieldDisc (d1, d2, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating purchase price (per $100 face value).
n2 – a numeric value indicating the security redemption value (per $100 face value).

YieldDisc (d1, d2, n1, n2, n3)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
n1 – a numeric value indicating purchase price (per $100 face value).
n2 – a numeric value indicating the security redemption value (per $100 face value).
n3 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
YieldDisc(#1/1/2000#,#1/1/2005#,36.25,50)
```

returns .0759 (when formatted with four decimal places).

YieldMat

Returns a numeric value indicating the yield of a security that pays interest only at maturity.

YieldMat (d1, d2, d3, n1, n2)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating purchase price (per $100 face value).

YieldMat (d1, d2, d3, n1, n2, n3)

d1 – a date or date-time value indicating the purchase date.
d2 – a date or date-time value indicating the date of maturity.
d3 – a date or date-time value indicating the issue date.
n1 – a numeric value indicating the interest rate.
n2 – a numeric value indicating purchase price (per $100 face value).

n3 – a numeric value indicating the day basis system: 0 = US (NASD) 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360.

```
YieldMat(#3/1/2000#,#3/1/2005#,#1/1/2000#,.055,101.82)
```

returns .05.

Functions: Math

This set of functions performs standard math, algebraic, and geometrical calculations.

Abs

Returns a numeric value indicating the absolute value of a number.

Abs (n)

n – a numeric value to return the absolute value for.

```
Abs(-27.3)
```

returns 27.30.

Atn

Returns a numeric value (in radians) indicating the arctangent of the source number.

Atn (n)

n – a numeric value to return the arctangent of.

```
Atn(45)
```

returns 1.55.

Cos

Returns a numeric value indicating the cosine of the source number.

Cos (n)

n – a numeric value, in radians, to return the cosine of.

```
Cos(45)
```

returns .53.

Exp

Returns a numeric value indicating *e* (the base of the natural logarithm) raised to a power.

Exp (n)

n – a numeric value indicating the power to raise *e* to.

```
Exp(10)
```

returns 22,026.47 (*e* raised to the power of 10).

Int

Returns a numeric value indicating the integer portion of the supplied number.

Int (n)

n – a numeric value that you want to extract the integer portion of.

```
Int(3.574)
```

returns 3.00.

Log

Returns a numeric value indicating the logarithm of the supplied number.

Log (n)

n – a numeric value to return the logarithm of.

```
Log(22026.47)
```

returns 10.

Pi

Returns the value of Pi (approximately 3.142).

Although this function appears as "Pi" in the Function tree, "crPi" will appear when you double-click the function name. If you type in the function yourself, type it as "crPi".

crPi

```
crPi
```

returns 3.142 (when formatted to show three decimal places).

Remainder

Returns a numeric value indicating the remainder of a division operation (not the quotient).

The Remainder function operates similar to the Mod arithmetic operator. Among other things, Remainder can be used with the RecordNumber function to shade every other line of a report in alternating colors.

Remainder (n1, n2)

n1 – a numeric value indicating the numerator of the division.
n2 – a numeric value indicating the denominator of the division.

```
Remainder (10,3)
```

returns 1.00 (the remainder of dividing 10 by 3).

```
If Remainder (RecordNumber, 2) = 0 Then
   crSilver
Else
   crNoColor
```

will shade every other details section silver if used as a conditional formatting formula for the details sections background color.

Rnd

Returns a numeric value between 0 and 1 represented as a random number.

Rnd may be helpful in situations where you wish to add a certain "randomness" to portions of your report. By multiplying Rnd by a larger number (10, 100, and so forth), you may get larger numbers that may be appropriate for other uses.

Rnd

Rnd (n)

n – a number to act as a "seed" to generate the random number.

```
Rnd
```

returns a different fractional number within each record and report refresh.

```
Rnd(Timer)
```

uses the Timer function to generate a "seed" number to increase "randomness" of the numbers returned.

Round

Returns a numeric value rounded to a specified number of decimal places.

Round (n)

n – a numeric value to be rounded to the nearest whole number.

Round (n1, n2)

n1 – a numeric value to be rounded.
n2 – a numeric value indicating the number of decimal places to round *n1* to.

```
Round (1.7821, 1)
```

returns 1.80.

Sgn

Returns a numeric value indicating the sign of the supplied number.

Sgn returns –1 if the source number is negative, 0 if the source number is zero, and 1 if the source number is positive.

Sgn (n)

n – a numeric value to determine the sign of.

```
Sgn (-25)
```

returns –1.00.

Sin

Returns a numeric value indicating the sine of the supplied angle.

Sin (n)

n – a numeric value indicating the angle to calculate the sine of (in radians).

```
Sin(45)
```

returns .85.

Sqr

Returns a numeric value indicating the square root of the source number.

Sqr (n)

n – a numeric value to return the square root of.

```
Sqr (25)
```

returns 5.00.

Tan

Returns a numeric value indicating the tangent of the supplied angle.

Tan (n)

n – a numeric value indicating an angle (in radians).

```
Tan(45)
```

returns 1.62.

Truncate

Returns a numeric value indicating the source number truncated to a specified number of decimal places.

Truncate is similar to Round in that it removes decimal places. However, Truncate doesn't round the source number in the process—it simply removes the decimal places.

Truncate (n)

n – a numeric value to be truncated to zero decimal places.

Truncate (n1, n2)

n1 – a numeric value to be truncated.
n2 – a numeric value indicating the number of decimal places to truncate.

```
Truncate (1.489,1)
```

returns 1.40.

Functions: Print State

This category of functions exposes a variety of functions that generally pertain to the current state of the report when being formatted or printed. Some of these functions duplicate items found in the Special Fields category of the Field Explorer.

DrillDownGroupLevel

Returns a numeric value indicating how many group levels a viewer has drilled down into when viewing the report online (0 is returned if a viewer has not drilled down).

This function is very helpful when you need to determine where a viewer is in a drill-down hierarchy. You may use this function, for example, to suppress a group header if a viewer has drilled down to the group's first level (showing the series of just group headers and footers for the group). You will see a different DrillDownGroupLevel result and be able to show the group header when the viewer drills down to the next level where they're seeing details sections and just a single group header and footer.

DrillDownGroupLevel

```
DrillDownGroupLevel
```

returns 2.00 if the viewer has drilled into the second group.

```
DrillDownGroupLevel = 1
```

will suppress Group Header #1 when a viewer first drills down when placed in a conditional suppression formula for Group Header #1.

GroupNumber

Returns a numeric value indicating the sequential number of the group that is currently being displayed.

GroupNumber is helpful in various conditional formatting situations, such as conditionally setting a New Page Before property, or shading every other group footer. This function returns the same value as the GroupNumber special field.

GroupNumber

```
GroupNumber > 1
```

when used as a conditional New Page Before formula for Group Header #1, will not start a new page before the report's first group.

GroupSelection

Returns a string value indicating the Group Selection Formula.

This function returns the same information as the Group Selection Formula special field.

GroupSelection

```
GroupSelection
```

returns "Sum ({Orders.Order Amount}, {Orders.Customer ID}) > $15000.00" if a group selection formula is limited to groups on the report.

InRepeatedGroupHeader

Returns a Boolean (true/false) value indicating whether or not the report is currently exposing a repeated group header.

InRepeatedGroupHeader is helpful when you wish to avoid resetting variables in a repeated group header, when you want to add a "continued" message, or similar conditional behavior.

InRepeatedGroupHeader

```
If Not InRepeatedGroupHeader Then
    NumberVar GroupTotal := 0
```

will reset the GroupTotal variable to 0 only if this is the first group header for the group.

IsNull

Returns a Boolean (true/false) value indicating if the supplied field contains a null value.

Null database values may not appear with all databases. Also, the setting of the Convert Null Database Fields To Default option (from File | Report Options or File | Options) affects whether null database values will exist or not.

IsNull (f)

f – any field name from a database table.

```
If IsNull ({Sales.Cust}) Then
   "No customer for this sale"
Else
   {Sales.Cust}
```

returns the string "No customer for this sale" if the field is null. Otherwise, it returns the field's value.

Next (Crystal Syntax only)

Returns the contents of the supplied field for the next database record.

Next (f)

f – any field name from a database table.

```
If {Sales.Cust} = Next({Sales.Cust}) Then
    {Sales.Cust} & " continues on next page"
```

compares the current customer name to the next customer name. If they're the same, the formula returns a "continues on next page" message.

NextIsNull

Returns a Boolean (true/false) value indicating if the supplied field in the next record contains a null value.

Null database values may not appear with all databases. Also, the setting of the Convert Null Database Fields To Default option (from File | Report Options or File | Options) affects whether null database values will exist or not.

NextIsNull (f)

f – any field name from a database table.

```
If NextIsNull ({Sales.Cust}) Then
   "Last order for " & {Sales.Cust}
```

returns a "last order" string if the next record's customer field is null (this assumes that the report is sorted on {Sales.Cust}).

NextValue (Basic Syntax only)

Returns the contents of the supplied field for the next database record.

This function is equivalent to Next. See **Next** for more information.

OnFirstRecord

Returns a Boolean (true/false) value indicating if the current record is the first record on the report.

OnFirstRecord is helpful for some conditional formatting requirements.

OnFirstRecord

```
Not OnFirstRecord
```

when used as a conditional formula for Group Header #1's New Page Before property, avoids a page break before the first group header on the report.

OnLastRecord

Returns a Boolean (true/false) value indicating if the current record is the last record on the report.

OnLastRecord is helpful for some conditional formatting requirements.

OnLastRecord

```
Not OnLastRecord
```

when used as a conditional formula for Group Footer #1's New Page After property, avoids a page break after the last group footer on the report.

PageNofM

Returns a string value indicating the current page number, followed by " of ", followed by the total number of report pages.

PageNofM returns the same string as the Page N of M special field.

PageNofM

```
PageNofM
```

returns "Page 3 of 20" if the current page being formatted is page 3 of a 20-page report.

PageNumber

Returns a numeric value indicating the current page being formatted or printed on the report.

PageNumber returns the same value as the Page Number special field.

PageNumber

```
PageNumber = 1
```

when supplied as a conditional formula for the Page Header Suppress property, will prevent the Page Header from being printed on the first page of the report.

Previous (Crystal Syntax only)

Returns the contents of the supplied field for the previous database record.

Previous (f)

f – any field name from a database table.

```
If {Sales.Cust} = Previous({Sales.Cust}) Then
    {Sales.Cust} & " continued from previous page"
```

compares the current customer name to the previous customer name. If they're the same, the formula returns a "continued from previous page" message.

PreviousIsNull

Returns a Boolean (true/false) value indicating if the supplied field in the previous record contains a null value.

Null database values may not appear with all databases. Also, the setting of the Convert Null Database Fields To Default option (from File | Report Options or File | Options) affects whether null database values will exist or not.

PreviousIsNull (f)

f – any field name from a database table.

```
If PreviousIsNull ({Sales.Cust}) Then
   "First order for " & {Sales.Cust}
```

returns a "first order" string if the previous record's customer field is null (this assumes that the report is sorted on {Sales.Cust}).

PreviousValue (Basic Syntax only)

Returns the contents of the supplied field for the previous database record.

This function is equivalent to Previous. See **Previous** for more information.

RecordNumber

Returns a numeric value indicating the sequential number of the current record.

RecordNumber can be used for certain conditional formatting situations. RecordNumber returns the same value as the Record Number special field.

RecordNumber

```
If RecordNumber Mod 2 = 0 Then
   crSilver
Else
   crNoColor
```

when supplied as a conditional formula to the Details section background color property, shades every other details section silver.

RecordSelection

Returns a string value containing the report's record selection formula.

RecordSelection returns the same value as the Record Selection Formula special field.

RecordSelection

```
RecordSelection
```

returns "{Sales.State} in ["MT", "ID", "WY", "CO"]" if a record selection formula has been created limiting the report to several states.

TotalPageCount

Returns a numeric value indicating the total number of pages on the report.

TotalPageCount returns the same value as the Total Page Count special field.

TotalPageCount

```
If PageNumber < TotalPageCount Then "report continues on next page"
```

prints a "report continues on next page" message of the report is not printing the last page.

Functions: Programming Shortcuts

This category of functions supplies shortcuts for other, more involved programming constructs (such as If-Then-Else or Select-Case).

Choose

Returns a value (the data type depends on function arguments) from the list of "choices."

Choose (n, c1, c2...)

n – a numeric value indicating which entry in the "choice" list to return.
c1 – the first entry in the choice list. *c1* can be any data type or a range value (arrays, however, aren't allowed).
c2 – the next entry in the choice list. It must be the same data type as *c1.* Additional choice entries can follow *c2,* each separated by a comma.

```
Choose(5,"January","February","March","April","May","June")
```

returns "May".

IIF

Returns a value (the data type depends on function arguments) based on the outcome of a conditional test.

IIF (e, v1, v2)

e – a Boolean expression using comparison operators or other Boolean logic. Returns True or False.
v1 – a value of any data type (excluding an array) indicating what the function returns if *e* is true.
v2 – a value of the same data type as *v1* indicating what the function returns if *e* is false.

```
IIF({Sales.Amount} > 5000,"Bonus Sale","Normal Sale")
```

returns the string "Bonus Sale" for sales over $5,000. Otherwise, returns the string "Normal Sale".

Switch

Returns a value (the data type depends on function arguments) based on the outcome of several conditional tests.

Switch (e1, v1, e2..., v2...)

e1 – a Boolean expression using comparison operators or other Boolean logic. Returns True or False.
v1 – a value of any data type (excluding an array) that's returned if *e1* evaluates to True.
e2 – a Boolean expression using comparison operators or other Boolean logic. Returns True or False.
v2 – a value of the same data type as *v1* that's returned if *e2* evaluates to True.
Additional pairs of expressions and values may be supplied. An equal number of expressions and values must be supplied.

```
Switch ({Sales.Amount} < 100, "Needs Improvement",
        {Sales.Amount} < 5000, "OK Sale",
        {Sales.Amount} < 7500, "Bonus Sale",
        True, "Fantastic Sale!")
```

returns a string value relating to the comparisons of the amount field. If the first three tests fail, the formula returns "Fantastic Sale".

Functions: Ranges

This category of functions test range values to determine if they contain a fixed lower or upper bound.

HasLowerBound

Returns a Boolean (true/false) value indicating whether the supplied range has a defined lower bound.

If you supply a range array to HasLowerBound, *every* element of the array must have a lower bound for HasLowerBound to return True.

HasLowerBound (r)

r – a range or range array.

```
If HasLowerBound ({?Order Amounts}) Then
    "Orders start at " & ToText(Minimum({?Order Amounts}))
```

returns "Orders start at 1,000.00" if the Order Amounts parameter field (defined as a range parameter field) has a lower bound of 1,000 specified.

HasUpperBound

Returns a Boolean (true/false) value indicating whether the supplied range has a defined upper bound.

If you supply a range array to HasUpperBound, *every* element of the array must have an upper bound for HasUpperBound to return True.

HasUpperBound (r)

r – a range or range array.

```
If HasUpperBound ({?Order Amounts}) Then
    "Orders end at " & ToText(Maximum({?Order Amounts}))
```

returns "Orders end at 5,000.00" if the Order Amounts parameter field (defined as a range parameter field) has an upper bound of 5,000 specified.

IncludesLowerBound

Returns a Boolean (true/false) value indicating whether the supplied range has a defined lower bound.

If you supply a range array to IncludesLowerBound, *any* element of the array may have a lower bound for IncludesLowerBound to return True.

IncludesLowerBound (r)

r – a range or range array.

```
If IncludesLowerBound ({?Order Amounts}) Then
    "Orders start at " & ToText(Minimum({?Order Amounts}))
```

returns "Orders start at 1,000.00" if the Order Amounts parameter field (defined as a range parameter field) has a lower bound of 1,000 specified.

IncludesUpperBound

Returns a Boolean (true/false) value indicating whether the supplied range has a defined upper bound.

If you supply a range array to IncludesUpperBound, *any* element of the array may have an upper bound for IncludesUpperBound to return True.

IncludesUpperBound (r)

r – a range or range array.

```
If IncludesUpperBound ({?Order Amounts}) Then
    "Orders end at " & ToText(Maximum({?Order Amounts}))
```

returns "Orders end at 5,000.00" if the Order Amounts parameter field (defined as a range parameter field) has an upper bound of 5,000 specified.

Functions: Strings

This category of functions allows manipulation or testing of string values, database fields, and other string formulas.

AscW

Returns a numeric value indicating the Unicode value of the first character of the supplied string.

AscW (s)

s – a string value.

```
AscW ("X")
```

returns 88.00 (the Unicode code for a capital X).

NOTE *AscW replaces Asc, which is still supported for backward compatibility. Asc behaves identically to AscW.*

ChrW

Returns a string value indicating the Unicode representation of the supplied number.

ChrW (n)

n – a whole number.

```
ChrW(88)
```

returns "X".

NOTE *ChrW replaces Chr, which is still supported for backward compatibility. Chr behaves identically to ChrW.*

Filter

Returns a string array containing only strings that match the search string.

Filter (sa, s)

sa – a string array containing one or more individual strings to search.
s – a string value to search *sa* for.

Filter (sa, s, b)

sa – a string array containing one or more individual strings to search.
s – a string value to search *sa* for.
b – a Boolean (true/false) value indicating whether the search should return partial matches. If True, partial string matches will be returned. If False, only complete matches will be returned.

Filter (sa, s, b, n)

sa – a string array containing one or more individual strings to search.
s – a string value to search *sa* for.

b – a Boolean (true/false) value indicating whether the search should return string matches or nonmatches. If True, strings that match *s* will be returned. If False, strings that *do not* match *s* will be returned.
n – a numeric value indicating whether the search is case-sensitive. 0 = case-sensitive search, 1 = non-case-sensitive search.

```
StringVar Array Beatles := ["John","Paul","George","Ringo"];
Join (Filter(Beatles,"o"))
```

returns "John George Ringo" (the Join function concatenates the three elements of the Beatles string array that contain the letter "o").

InStr

Returns a numeric value indicating the first position in the source string where the search string appears.

InStr (s1, s2)

s1 – a string value indicating the source string to search.
s2 – a string value indicating the string to search for in *s1.*

InStr (n, s1, s2)

n – a numeric value indicating the position in *s1* to start searching for *s2.*
s1 – a string value indicating the source string to search.
s2 – a string value indicating the string to search for in *s1.*

InStr (s1, s2, n)

s1 – a string value indicating the source string to search.
s2 – a string value indicating the string to search for in *s1.*
n – a numeric value indicating whether the search is case-sensitive: 0 = case-sensitive, 1 = not case-sensitive.

InStr (n1, s1, s2, n2)

n1 – a numeric value indicating the position in *s1* to start searching for *s2.*
s1 – a string value indicating the source string to search.
s2 – a string value indicating the string to search for in *s1.*
n2 – a numeric value indicating whether the search is case-sensitive: 0 = case-sensitive, 1 = not case-sensitive.

```
Instr("The rain in Spain falls on the plain","spain")
```

returns 0.00 (the search is case-sensitive).

```
Instr("The rain in Spain falls on the plain","spain",1)
```

returns 13.00 (the search is not case-sensitive).

InStrRev

Returns a numeric value indicating the last position in the source string where the search string appears.

In essence, InStrRev starts searching for the substring from right to left.

InStrRev (s1, s2)

s1 – a string value indicating the source string to search.
s2 – a string value indicating the string to search for in *s1.*

InStrRev (s1, s2, n)

s1 – a string value indicating the source string to search.
s2 – a string value indicating the string to search for in *s1.*
n – a numeric value indicating the position from the right of *s1* to start searching for *s2.*

InStrRev (s1, s2, n1, n2)

s1 – a string value indicating the source string to search.
s2 – a string value indicating the string to search for in *s1.*
n1 – a numeric value indicating the position from the right of *s1* to start searching for *s2.*
n2 – a numeric value indicating whether the search is case-sensitive: 0 = case-sensitive, 1 = not case-sensitive.

```
InstrRev("The rain in Spain falls on the plain","a",25)
```

returns 20.00 (the search starts at position 25 and proceeds left—finding a lowercase "a" at position 20).

Join

Returns a string value consisting of all the elements of the source string array concatenated together.

Join is helpful for, among other things, displaying all the values entered into a multivalue string parameter field in a single formula.

Join (a)

a – a string array.

Join (a, s)

a – a string array.
s – a string value to act as a "delimiter"; it will be inserted between each array element in the output string.

```
Join({?States Chosen},", ")
```

returns "CO, WY, MT, UT" if the four states were supplied to the multivalue States Chosen parameter field.

Left

Returns a string value indicating a limited number of characters extracted from the left side of the source string.

Left (s, n)

s – a string value used as the source string.
n – a numeric value indicating how many characters to extract from *s*.

```
Left ("George", 3)
```

returns "Geo".

Length

Returns a numeric value indicating how many characters are contained in the source string.
Len is functionally equivalent to Length.

Length (s)

s – a string value.

```
Length ("George")
```

returns 6.00.

LowerCase

Returns a string value indicating a lowercase representation of the source string.
LCase is functionally equivalent to LowerCase.

LowerCase (s)

s – a string value to be converted to lowercase.

```
LowerCase ("George Peck")
```

returns "george peck".

Mid

Returns a string value indicating a substring extracted from the source string.

Mid (s, n)

s – a string value to extract characters from.
n – a numeric value indicating the position to begin the extraction from *s*.

Mid (s, n1, n2)

s – a string value to extract characters from.
n1 – a numeric value indicating the position to begin the extraction from *s*.
n2 – a numeric value indicating how many characters to extract.

```
Mid("George Peck",8,3)
```

returns "Pec".

NumericText

Returns a Boolean (true/false) value indicating whether or not the source string contains only numeric characters.

NumericText is helpful for testing the contents of a string value before converting it with the ToNumber function, as ToNumber will fail if it encounters any nonnumeric characters.

NumericText (s)

s – a string value to test for numeric content.

```
If NumericText ({FromMainframe.SalesAmount}) Then
   ToNumber({FromMainframe.SalesAmount}) * 1.1
Else
   0
```

checks that the SalesAmount string field contains only numeric values before converting it to a number with the ToNumber function.

ProperCase

Returns a string value converted so that the first character, and any character immediately appearing after a space or nonalphanumeric character, will appear in uppercase. The remaining characters will appear in lowercase.

This function is very helpful when encountering proper names, such as first and last names, that are stored in a database in all uppercase characters.

ProperCase (s)

s – a string value.

```
ProperCase("GEORGE PECK,JR.")
```

returns "George Peck,Jr."

Replace

Returns a string value where a certain set of characters have been replaced by another set of characters.

Replace (s1, s2, s3)

s1 – the source string to search.
s2 – the characters to search *s1* for.
s3 – the characters to replace *s2* with in *s1.*

Replace (s1, s2, s3, n)

s1 – the source string to search.
s2 – the characters to search *s1* for.

s3 – the characters to replace *s2* with in *s1.*
n – a numeric value indicating the position in *s1* to start searching for *s2.*

Replace (s1, s2, s3, n1, n2)

s1 – the source string to search.
s2 – the characters to search *s1* for.
s3 – the characters to replace *s2* with in *s1.*
n1 – a numeric value indicating the position in *s1* to start searching for *s2.*
n2 – a numeric value indicating the number of replacements to make.

Replace (s1, s2, s3, n1, n2, n3)

s1 – the source string to search.
s2 – the characters to search *s1* for.
s3 – the characters to replace *s2* with in *s1.*
n1 – a numeric value indicating the position in *s1* to start searching for *s2.*
n2 – a numeric value indicating the number of replacements to make.
n3 – a numeric value indicating whether the search is case-sensitive: 0 = case-sensitive, 1 = not case-sensitive.

```
Replace("George Peck","George","Gregory")
```

returns "Gregory Peck".

ReplicateString

Returns a string value consisting of multiple duplications of a source string.

ReplicateString (s, n)

s – the source string to duplicate.
n – a numeric value indicating how many duplications of *s* to make.

```
ReplicateString ("*", 15)
```

returns "***************".

Right

Returns a string value indicating a limited number of characters extracted from the right side of the source string.

Right (s, n)

s – a string value used as the source string.
n – a numeric value indicating how many characters to extract from *s*.

```
Right ("George", 3)
```

returns "rge".

Roman

Returns a string value expressing the supplied numeric argument as a Roman numeral value.

Roman (n)

n – a numeric value to convert to Roman numerals.

Roman (n1, n2)

n1 – a numeric value to convert to Roman numerals.
n2 – a numeric value indicating the type of Roman numeral to return; 0 = Classic; 1, 2, or 3 = various levels of precision; 4 = simplified.

```
Roman (1998)
```

returns "MCMXCVIII".

```
Roman (1998, 2)
```

returns "MXMVIII".

NOTE *The best way to explore the various precision choices is to simply try them.*

Space

Returns a string value consisting of a specified number of spaces.

Space (n)

n – the number of space characters to duplicate.

```
"George" & Space(10) & "Peck"
```

returns "George Peck".

Split

Returns a string array consisting of elements built from the supplied individual strings.

Split (s)

s – a string containing one or more individual strings separated by spaces.

Split (s1, s2)

s1 – a string containing one or more individual strings separated by the delimiter specified in *s2.*
s2 – a string value indicating the delimiter that separates individual values in *s1.*

Split (s1, s2, n)

s1 – a string containing one or more individual strings separated by the delimiter specified in *s2.*
s2 – a string value indicating the delimiter that separates individual values in *s1.*
n – a numeric value indicating how many elements should be placed in the array.

Split (s1, s2, n1, n2)

s1 – a string containing one or more individual strings separated by the delimiter specified in *s2.*
s2 – a string value indicating the delimiter that separates individual values in *s1.*
n1 – a numeric value indicating how many elements should be placed in the array (–1 for all).
n2 – a numeric value indicating how to treat the search for the delimiter. 0 = search is case-sensitive, 1 = search is not case-sensitive.

```
Split("John Paul George Ringo")[2]
```

returns "Paul".

```
Split("John And Paul And George And Ringo","and",-1,0)[1]
```

returns "John And Paul And George And Ringo", as only one element was added to the array—the delimiter search is case-sensitive and "and" is not found.

```
Split("John And Paul And George And Ringo","and",-1,1)[1]
```

returns "John"—the delimiter search is not case-sensitive.

StrCmp

Returns a numeric value indicating how two strings compare alphabetically.

StrCmp returns –1 if the first string is less than the second string, 0 if the strings are equal, and 1 if the first string is greater than the second string.

StrCmp (s1, s2)

s1 – the first string to compare.
s2 – the string to compare to *s1.*

StrCmp (s1, s2, n)

s1 – the first string to compare.
s2 – the string to compare to *s1.*
n – a numeric value indicating how to treat the search for the delimiter. 0 = search is case-sensitive, 1 = search is not case-sensitive.

```
StrCmp ("abc","xyz")
```

returns –1.00.

StrReverse

Returns a string value consisting of the source string in reverse order.

StrReverse (s)

s – a string value to be reversed.

```
StrReverse ("George Peck")
```

returns "kceP egroeG".

ToNumber

Returns a numeric value indicating a numeric conversion of another nonnumeric value.

ToNumber converts other numeric and Boolean values, as well as string values, to numbers. ToNumber is functionally equivalent to CDbl and Val.

ToNumber (v)

v – a numeric, string, or Boolean value to be converted to a number.

```
ToNumber ("5642.15")
```

returns 5,642.15 as a number.

NOTE *You may wish to test the value to convert with NumericText before you supply it to ToNumber, as ToNumber will fail if a string is supplied that doesn't contain all numeric values. If you wish to avoid the test, you may use Val, which will always return a result (perhaps 0), even if the supplied argument isn't numeric.*

ToText

Converts any nonstring data type to a string value.

ToText is an immensely valuable function used to convert any nonstring data type to a string. You'll often use ToText within a string formula to concatenate various types of data items into a single string result.

ToText (b)

b – a Boolean (true/false) value to be converted to a string.

```
ToText({Orders.Shipped})
```

returns the string "True" if the Boolean Shipped field holds a true value or "False" if the field is false.

ToText (n1, n2, s1, s2)

n1 – a numeric value to be converted to a string.
n2 – a numeric value indicating the number of decimal places to use when converting *n1.* This argument is optional.

s1 – a string value indicating the character or characters to use as a thousands separator when converting *n1*. This argument is optional.
s2 – a string value indicating the character or characters to use as a decimal separator when converting *n1*. This argument is optional.

```
ToText(21256.1255, 2, ",", "-")
```

returns the string "21,256-13" (showing two decimal places, a comma thousands separator, and a dash decimal separator).

ToText (n1, n2, s1, s2, s3)

n1 – a numeric value to be converted to a string.
s1 – a string value used to format the number. This argument is optional.
n2 – a numeric value indicating the number of decimal places to use when converting *n1*. This argument is optional.
s2 – a string value indicating the character or characters to use as a thousands separator when converting *n*. This argument is optional.
s3 – a string value indicating the character or characters to use as a decimal separator when converting *n*. This argument is optional.

```
ToText(21256.1255, "#,###,###.00000",2)
```

returns " 21,256.13000" (with leading spaces for extra # characters).

Numeric Formatting Strings

Character	Usage
#	Numeric placeholder. A single numeric character will be placed in each occurrence of #. If there are more # characters than numeric digits, leading or trailing spaces will be added for extra # characters.
0 (zero)	Numeric placeholder. A single numeric character will be placed in each occurrence of 0. If there are more 0 characters than numeric digits, leading or trailing zeros will be added for extra 0 characters.
, (comma)	Thousands separator. Indicates where the thousands separator should be placed in the resulting string. The thousands separator used is taken from a Windows default or the *s2* argument.
. (period)	Decimal separator. Indicates where the decimal separator should be placed in the resulting string. The decimal separator used is taken from a Windows default or the *s2* argument.

ToText (d, s1, s2, s3)

d – a date, time, or date/time value.
s1 – a string value used to format the date, time, or date/time value. This argument is optional.
s2 – a string value used as a replacement label for "a.m." for morning hours. This argument is optional.
s3 – a string value used as a replacement label for "p.m." for afternoon/evening hours. This argument is optional.

```
ToText(CurrentDate, "dddd M d, yyyy")
```

returns "Friday May 9, 2003" if CurrentDate contains May 9, 2003.

```
ToText(CurrentTime, "h:mm tt",
"in the morning", "in the afternoon")
```

returns the string " 4:27 in the afternoon" if CurrentTime is 4:27 P.M.

NOTE *Any nonrecognized characters in the format string will be returned as a literal character. For example, a slash (/) or comma (,) in the format string will simply display within the resulting date string.*

Date/Time Formatting Strings

Character	Usage
M	Month number without a leading zero for single-character month
MM	Month number with a leading zero for single-character month
MMM	Month name as a three-letter abbreviation
MMMM	Month name fully spelled
d	Day of month without a leading zero for single-character day
dd	Day of month with a leading zero for single-character day
ddd	Day of week name spelled as a three-letter abbreviation
dddd	Day of week name fully spelled
yy	Last two characters of year
yyyy	Full four-character year
h	Hours without a leading zero for a single-character hour (12-hour nonmilitary format)
hh	Hours with a leading zero for a single-character hour (12-hour nonmilitary format)
H	Hours without a leading zero for a single-character hour (24-hour military format)
HH	Hours with a leading zero for a single-character hour (24-hour military format)
m	Minutes without a leading zero for a single-character minute
mm	Minutes with a leading zero for a single-character minute
s	Seconds without a leading zero for a single-character second
ss	Seconds with a leading zero for a single-character second
t	Single-character uppercase A or P (for A.M. or P.M.)
tt	Multiple-character uppercase AM or PM (for A.M. or P.M.)

NOTE *CStr is functionally equivalent to ToText. It is provided to maintain compatibility with Basic syntax.*

ToWords

Returns a string value indicating the "spelled out" version of the numeric argument.

ToWords is often used to print checks, but can also be used when required to convert numeric values to text for other situations. For example, ToWords will be helpful if you are required to create a formula, based on numeric data, that returns the string "This contract expires in 30 (thirty) days".

ToWords (n)

n – a numeric value to spell as words.

```
ToWords (1145.31)
```

returns the string "one thousand one hundred forty-five and 31 / 100".

ToWords (n1, n2)

n1 – a numeric value to spell as words.
n2 – a numeric value indicating the number of decimal places to use when converting *n1.*

```
ToWords (100, 0)
```

returns the string "one hundred".

NOTE *An optional third numeric argument specifying the form type can be supplied. Allowable values for the form type argument are numeric and can be 0 (Classic or Check form), 1 (Daily form), or 2 (Casual form). This argument applies only to Asian language versions of Crystal Reports.*

Trim

Returns a string value consisting of the source string with all leading and trailing spaces removed.

Trim (s)

s – a string value.

```
Trim ("    Leading and trailing spaces   ")
```

returns the string "Leading and trailing spaces" with no extra spaces before or after the string.

TrimLeft

Returns a string value consisting of the source string with all leading spaces removed.

LTrim is functionally equivalent to TrimLeft.

TrimLeft (s)

s – a string value.

```
TrimLeft ("    Leading and trailing spaces   ")
```

returns the string "Leading and trailing spaces " with no extra spaces before the string, but with any trailing spaces remaining.

TrimRight

Returns a string value consisting of the source string with all trailing spaces removed.
RTrim is functionally equivalent to TrimRight.

TrimRight (s)

s – a string value.

```
TrimRight ("    Leading and trailing spaces   ")
```

returns the string " Leading and trailing spaces" with no extra spaces after the string, but with any leading spaces remaining.

UpperCase

Returns a string value indicating an uppercase representation of the source string.
UCase is functionally equivalent to UpperCase.

UpperCase (s)

s – a string value to be converted to uppercase.

```
UpperCase ("George Peck")
```

returns "GEORGE PECK".

Val

Returns a numeric value indicating the numeric portion of the supplied string.
Val is helpful for converting string database fields that contain numeric data to actual numeric values. In particular, Val will not return an error if it encounters a nonnumeric character in the string—it will simply stop the conversion at that point.

Val (s)

s – a string value to be converted to a number.

```
Val ("12500.1a31")
```

returns 12,500.10 as a numeric value.

Functions: Summary

This category of functions largely duplicates summary functions available via the Insert | Summary option from the Crystal Reports pull-down menus. By using these summary functions, you may include the same summary calculations available from the Insert | Summary option in formulas. In fact, if you add a report summary to a report with Insert | Summary and then double-click it in the Fields tree of the Formula Editor, the following summary functions will be inserted in your report automatically.

Average

Returns a numeric value indicating the average of the supplied database field or formula.

Average (f)
f – the database or formula field to summarize for the entire report. This must be a numeric field.

Average (f1, f2)
f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Average (f1, f2, s)
f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Average({Sales.Amount})
```

returns the average of sales amounts for the entire report.

```
Average({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the average of sales amounts for the sales date group, summarized for each month.

Correlation
Returns a numeric value indicating the correlation between the supplied database fields or formulas.

Correlation (f1, f2)
f1 – the first database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the second database or formula field to summarize for the entire report. This must be a numeric field.

Correlation (f1, f2, f3)
f1 – the first database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the second database or formula field to summarize for the entire report. This must be a numeric field.
f3 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Correlation (f1, f2, f3, s)
f1 – the first database or formula field to summarize for the entire report. This must be a numeric field.

f2 – the second database or formula field to summarize for the entire report. This must be a numeric field.
f3 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Correlation({Sales.Amount}, {Sales.Goal})
```

returns the correlation between sales amount and sales goal for the entire report.

```
Correlation({Sales.Amount}, {Sales.Goal}, {Sales.Date}, "monthly")
```

returns the correlation between sales amount and sales goal for the sales date group, summarized for each month.

Count

Returns a numeric value indicating the number of occurrences (count) of the supplied database field or formula.

Count (f)

f – the database or formula field to summarize for the entire report.

Count (f1, f2)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Count (f1, f2, s)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Count({Sales.Amount})
```

returns the count of sales amounts for the entire report (in essence, this is the number of records on the report).

```
Count({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the count of sales amounts for the sales date group, summarized for each month (in essence, this is the number of records in the group).

NOTE *The Count function will count every record where the supplied field contains a nonnull value. If any occurrences of the supplied field contain null values, those records won't increment the count.*

Covariance

Returns a numeric value indicating the covariance between the supplied database fields or formulas.

Covariance (f1, f2)

f1 – the first database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the second database or formula field to summarize for the entire report. This must be a numeric field.

Covariance (f1, f2, f3)

f1 – the first database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the second database or formula field to summarize for the entire report. This must be a numeric field.
f3 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Covariance (f1, f2, f3, s)

f1 – the first database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the second database or formula field to summarize for the entire report. This must be a numeric field.
f3 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Covariance({Sales.Amount}, {Sales.Goal})
```

returns the covariance between sales amount and sales goal for the entire report.

```
Covariance({Sales.Amount}, {Sales.Goal}, {Sales.Date}, "monthly")
```

returns the covariance between sales amount and sales goal for the sales date group, summarized for each month.

DistinctCount

Returns a numeric value indicating the unique number of occurrences (distinct count) of the supplied database field or formula.

DistinctCount (f)

f – the database or formula field to summarize for the entire report.

DistinctCount (f1, f2)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

DistinctCount (f1, f2, s)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Distinctcount({Sales.Acct#})
```

returns the number of unique accounts for the entire report.

```
Distinctcount({Sales.Acct#}, {Sales.Date}, "monthly")
```

returns the number of unique accounts for the sales date group, summarized for each month.

Maximum

Returns a numeric value indicating the maximum of the supplied database field or formula.
Maximum returns the largest number, the latest date, or the string value last in the alphabet.

Maximum (f)

f – the database or formula field to summarize for the entire report.

Maximum (f1, f2)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Maximum (f1, f2, s)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Maximum({Sales.Amount})
```

returns the highest sales amount for the entire report.

```
Maximum({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the highest sales amount for the sales date group, summarized for each month.

Median

Returns a numeric value indicating the median of the supplied database field or formula.

Median (f)

f – the database or formula field to summarize for the entire report. This must be a numeric field.

Median (f1, f2)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Median (f1, f2, s)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Median({Sales.Amount})
```

returns the median of sales amounts for the entire report.

```
Median({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the median of sales amounts for the sales date group, summarized for each month.

Minimum

Returns a numeric value indicating the minimum of the supplied database field or formula.

Minimum returns the smallest number, the earliest date, or the string value first in the alphabet.

Minimum (f)

f – the database or formula field to summarize for the entire report.

Minimum (f1, f2)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Minimum (f1, f2, s)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Minimum({Sales.Amount})
```

returns the lowest sales amount for the entire report.

```
Minimum({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the lowest sales amount for the sales date group, summarized for each month.

Mode

Returns a numeric value indicating the mode (the most frequently occurring value) of the supplied database field or formula.

Mode (f)

f – the database or formula field to summarize for the entire report.

Mode (f1, f2)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Mode (f1, f2, s)

f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Mode({Sales.Acct#})
```

returns the most frequently appearing account number for the entire report.

```
Mode({Sales.Acct#}, {Sales.Date}, "monthly")
```

returns the most frequently appearing account number for the sales date group, summarized for each month.

NthLargest

Returns a numeric value indicating the nth largest occurrence of the supplied database field or formula.

NthLargest returns the "nth" largest number, the "nth" latest date, or the "nth" string value last in the ASCII sort order.

NthLargest (n, f)

n – a numeric value (between 1 and 100). 5 returns the fifth largest, 2 the second largest, and so forth.
f – the database or formula field to summarize for the entire report.

NthLargest (n, f1, f2)

n – a numeric value (between 1 and 100). 5 returns the fifth largest, 2 the second largest, and so forth.
f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

NthLargest (n, f1, f2, s)

n – a numeric value (between 1 and 100). 5 returns the fifth largest, 2 the second largest, and so forth.
f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
NthLargest(5, {Sales.Amount})
```

returns the fifth largest sales amount for the entire report.

```
NthLargest(5, {Sales.Amount}, {Sales.Date}, "monthly")
```

returns the fifth largest sales amount for the sales date group, summarized for each month.

NthMostFrequent

Returns a numeric value indicating the nth most frequent occurrence of the supplied database field or formula.

NthMostFrequent (n, f)

n – a numeric value (between 1 and 100). 5 returns the fifth most frequent, 2 the second most frequent, and so forth.
f – the database or formula field to summarize for the entire report.

NthMostFrequent (n, f1, f2)

n – a numeric value (between 1 and 100). 5 returns the fifth most frequent, 2 the second most frequent, and so forth.
f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

NthMostFrequent (n, f1, f2, s)

n – a numeric value (between 1 and 100). 5 returns the fifth most frequent, 2 the second most frequent, and so forth.
f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
NthMostFrequent(5, {Sales.Acct#})
```

returns the fifth most frequently occurring account number for the entire report.

```
NthMostFrequent(5, {Sales.Acct#}, {Sales.Date}, "monthly")
```

returns the fifth most frequently occurring account number for the sales date group, summarized for each month.

NthSmallest

Returns a numeric value indicating the nth smallest occurrence of the supplied database field or formula.

NthSmallest returns the "nth" smallest number, the "nth" earliest date, or the "nth" string value first in the ASCII sort order.

NthSmallest (n, f)

n – a numeric value (between 1 and 100). 5 returns the fifth smallest, 2 the second smallest, and so forth.
f – the database or formula field to summarize for the entire report.

NthSmallest (n, f1, f2)

n – a numeric value (between 1 and 100). 5 returns the fifth smallest, 2 the second smallest, and so forth.
f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

NthSmallest (n, f1, f2, s)

n – a numeric value (between 1 and 100). 5 returns the fifth smallest, 2 the second smallest, and so forth.
f1 – the database or formula field to summarize for a report group.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
NthSmallest (5, {Sales.Amount})
```

returns the fifth smallest sales amount for the entire report.

```
NthSmallest (5, {Sales.Amount}, {Sales.Date}, "monthly")
```

returns the fifth smallest sales amount for the sales date group, summarized for each month.

PercentOfAverage

Returns a numeric value indicating what percentage an average calculation in one group is of another average calculation in a later group or the entire report.

This function duplicates a Percentage Summary Field created with the Insert | Summary menu option.

PercentOfAverage (f1, f2, s)

This will show the percentage of the group value compared to the value for the entire report.
f1 – the database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument is required only if *f2* is a date, date-time, time, or Boolean field.

PercentOfAverage (f1, f2, s1, f3, s2)

This will show the percentage of the group value compared to the value for a later group.
f1 – the database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s1 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f2* is a date, date-time, time, or Boolean field.
f3 – the database or formula field indicating the later group field to use for the summary. An existing group on the report must be based on this field.
s2 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f3* is a date, date-time, time, or Boolean field.

```
PercentOfAverage ({Sales.Amount}, {Sales.CustName})
```

returns the percentage that the average sales amount for each customer group makes up of the average sales amount for the entire report.

```
PercentOfAverage ({Sales.Amount}, {Sales.Date},
"monthly", {Sales.CustName})
```

returns the percentage that the average sales amount for each month makes up of the average sales amount for the higher-level customer name group.

PercentOfCount

Returns a numeric value indicating what percentage a count calculation in one group is of another count calculation in a later group or the entire report.

This function duplicates a Percentage Summary Field created with the Insert | Summary menu option.

PercentOfCount (f1, f2, s)

This will show the percentage of the group value compared to the value for the entire report.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument is required only if *f2* is a date, date-time, time, or Boolean field.

PercentOfCount (f1, f2, s1, f3, s2)

This will show the percentage of the group value compared to the value for a later group.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s1 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f2* is a date, date-time, time, or Boolean field.
f3 – the database or formula field indicating the later group field to use for the summary. An existing group on the report must be based on this field.
s2 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f3* is a date, date-time, time, or Boolean field.

```
PercentOfCount ({Sales.Amount}, {Sales.CustName})
```

returns the percentage that the number of records for each customer group makes up of the total number of records for the entire report.

```
PercentOfCount ({Sales.Amount}, {Sales.Date},
"monthly", {Sales.CustName})
```

returns the percentage that the number of records for each month makes up of the number of records for the higher-level customer name group.

NOTE *PercentOfCount calculations will exclude records that contain a null value in the field being counted.*

PercentOfDistinctCount

Returns a numeric value indicating what percentage a distinct count calculation in one group is of another distinct count calculation in a later group or the entire report.

This function duplicates a Percentage Summary Field created with the Insert | Summary menu option.

PercentOfDistinctCount (f1, f2, s)

This will show the percentage of the group value compared to the value for the entire report.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument is required only if *f2* is a date, date-time, time, or Boolean field.

PercentOfDistinctCount (f1, f2, s1, f3, s2)

This will show the percentage of the group value compared to the value for a later group.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s1 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f2* is a date, date-time, time, or Boolean field.
f3 – the database or formula field indicating the later group field to use for the summary. An existing group on the report must be based on this field.
s2 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f3* is a date, date-time, time, or Boolean field.

```
PercentOfDistinctCount ({Sales.Acct#}, {Sales.CustName})
```

returns the percentage that the unique number of account numbers for each customer group makes up of the unique number of account numbers for the entire report.

```
PercentOfDistinctCount ({Sales.Acct#}, {Sales.Date},
"monthly", {Sales.CustName})
```

returns the percentage that the unique number of account numbers for each month makes up of the unique number of account numbers for the higher-level customer name group.

PercentOfMaximum

Returns a numeric value indicating what percentage a maximum calculation in one group is of another maximum calculation in a later group or the entire report.

This function duplicates a Percentage Summary Field created with the Insert | Summary menu option.

PercentOfMaximum (f1, f2, s)

This will show the percentage of the group value compared to the value for the entire report.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument is required only if *f2* is a date, date-time, time, or Boolean field.

PercentOfMaximum (f1, f2, s1, f3, s2)

This will show the percentage of the group value compared to the value for a later group.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s1 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f2* is a date, date-time, time, or Boolean field.
f3 – the database or formula field indicating the later group field to use for the summary. An existing group on the report must be based on this field.
s2 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f3* is a date, date-time, time, or Boolean field.

```
PercentOfMaximum ({Sales.Amount}, {Sales.CustName})
```

returns the percentage that the highest sales amount for each customer group makes up of the highest sales amount for the entire report.

```
PercentOfMaximum ({Sales.Amount}, {Sales.Date},
"monthly", {Sales.CustName})
```

returns the percentage that the highest sales amount for each month makes up of the highest sales amount for the higher-level customer name group.

PercentOfMinimum

Returns a numeric value indicating what percentage a minimum calculation in one group is of another minimum calculation in a later group or the entire report.

This function duplicates a Percentage Summary Field created with the Insert | Summary menu option.

PercentOfMinimum (f1, f2, s)

This will show the percentage of the group value compared to the value for the entire report.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument is required only if *f2* is a date, date-time, time, or Boolean field.

PercentOfMinimum (f1, f2, s1, f3, s2)

This will show the percentage of the group value compared to the value for a later group.
f1 – the database or formula field to summarize for the entire report.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s1 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f2* is a date, date-time, time, or Boolean field.
f3 – the database or formula field indicating the later group field to use for the summary. An existing group on the report must be based on this field.
s2 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f3* is a date, date-time, time, or Boolean field.

```
PercentOfMinimum ({Sales.Amount}, {Sales.CustName})
```

returns the percentage that the lowest sales amount for each customer group makes up of the lowest sales amount for the entire report.

```
PercentOfMinimum ({Sales.Amount}, {Sales.Date},
"monthly", {Sales.CustName})
```

returns the percentage that the lowest sales amount for each month makes up of the lowest sales amount for the higher-level customer name group.

PercentOfSum

Returns a numeric value indicating what percentage a sum calculation in one group is of another sum calculation in a later group or the entire report.

This function duplicates a Percentage Summary Field created with the Insert | Summary menu option.

PercentOfSum (f1, f2, s)

This will show the percentage of the group value compared to the value for the entire report.
f1 – the database or formula field to summarize for the entire report. This must be a numeric field.

f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument is required only if *f2* is a date, date-time, time, or Boolean field.

PercentOfSum (f1, f2, s1, f3, s2)

This will show the percentage of the group value compared to the value for a later group.
f1 – the database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s1 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f2* is a date, date-time, time, or Boolean field.
f3 – the database or formula field indicating the later group field to use for the summary. An existing group on the report must be based on this field.
s2 – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices. This argument (and the comma that precedes it) is required only if *f3* is a date, date-time, time, or Boolean field.

```
PercentOfSum ({Sales.Amount}, {Sales.CustName})
```

returns the percentage that the total sales amount for each customer group makes up of the total sales amount for the entire report.

```
PercentOfSum ({Sales.Amount}, {Sales.Date},
"monthly", {Sales.CustName})
```

returns the percentage that the total sales amount for each month makes up of the total sales amount for the higher-level customer name group.

PopulationStdDev

Returns a numeric value indicating the population standard deviation of the supplied database field or formula.

PopulationStdDev (f)

f – the database or formula field to summarize for the entire report. This must be a numeric field.

PopulationStdDev (f1, f2)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

PopulationStdDev (f1, f2, s)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
PopulationStdDev({Sales.Amount})
```

returns the population standard deviation of sales amounts for the entire report.

```
PopulationStdDev({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the population standard deviation of sales amounts for the sales date group, summarized for each month.

PopulationVariance

Returns a numeric value indicating the population variance of the supplied database field or formula.

PopulationVariance (f)

f – the database or formula field to summarize for the entire report. This must be a numeric field.

PopulationVariance (f1, f2)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

PopulationVariance (f1, f2, s)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
PopulationVariance({Sales.Amount})
```

returns the population variance of sales amounts for the entire report.

```
PopulationVariance({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the population variance of sales amounts for the sales date group, summarized for each month.

PthPercentile

Returns a numeric value indicating the pth percentile of the supplied database field or formula.

PthPercentile returns the "pth" percentile value, such as the value that equates to the tenth percentile of the report or group.

PthPercentile (n, f)

n – a numeric value (between 0 and 100). 5 returns the fifth percentile, 2 the second percentile, and so forth.
f – the database or formula field to summarize for the entire report. This must be a numeric field.

PthPercentile (n, f1, f2)

n – a numeric value (between 0 and 100). 5 returns the fifth percentile, 2 the second percentile, and so forth.
f1 – the database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

PthPercentile (n, f1, f2, s)

n – a numeric value (between 0 and 100). 5 returns the fifth percentile, 2 the second percentile, and so forth.
f1 – the database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
PthPercentile (5, {Sales.Amount})
```

returns the sales amount equating to the fifth percentile for the entire report.

```
PthPercentile (5, {Sales.Amount}, {Sales.Date}, "monthly")
```

returns the sales amount equating to the fifth percentile for the sales date group, summarized for each month.

StdDev

Returns a numeric value indicating the standard deviation of the supplied database field or formula.

StdDev (f)

f – the database or formula field to summarize for the entire report. This must be a numeric field.

StdDev (f1, f2)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

StdDev (f1, f2, s)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
StdDev({Sales.Amount})
```

returns the standard deviation of sales amounts for the entire report.

```
StdDev({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the standard deviation of sales amounts for the sales date group, summarized for each month.

Sum

Returns a numeric value indicating the sum of the supplied database field or formula.

Sum (f)

f – the database or formula field to summarize for the entire report. This must be a numeric field.

Sum (f1, f2)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Sum (f1, f2, s)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Sum({Sales.Amount})
```

returns the sum of sales amounts for the entire report.

```
Sum({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the sum of sales amounts for the sales date group, summarized for each month.

Variance

Returns a numeric value indicating the variance of the supplied database field or formula.

Variance (f)

f – the database or formula field to summarize for the entire report. This must be a numeric field.

Variance (f1, f2)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

Variance (f1, f2, s)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
Variance({Sales.Amount})
```

returns the variance of sales amounts for the entire report.

```
Variance({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the variance of sales amounts for the sales date group, summarized for each month.

WeightedAverage

Returns a numeric value indicating the average of the supplied database field or formula, given a weight by another field or formula.

WeightedAverage (f1, f2)

f1 – the database or formula field to summarize for the entire report. This must be a numeric field.
f2 – the database or formula field to use as the weight.

WeightedAverage (f1, f2, f3)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field to use as the weight.
f3 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.

WeightedAverage (f1, f2, f3, s)

f1 – the database or formula field to summarize for a report group. This must be a numeric field.
f2 – the database or formula field to use as the weight.
f3 – the database or formula field indicating the group you wish to summarize for. An existing group on the report must be based on this field.
s – a string value indicating how often to "change" the summary for Boolean, date, or date-time grouping. See **Boolean Conditions**, **Date Conditions**, and **Time Conditions** later in this appendix for available choices.

```
WeightedAverage({Sales.Amount},{Sales.Goal})
```

returns the average of sales amounts weighted by goal for the entire report.

```
WeightedAverage({Sales.Amount}, {Sales.Date}, "monthly")
```

returns the average of sales amounts weighted by goal for the sales date group, summarized for each month.

Boolean Conditions

The following string values can be used in a summary function when using a Boolean group field.

"any change"
"change to No"
"change to Yes"
"every No"
"every Yes"
"next is No"
"next is Yes"

Date Conditions

The following string values can be used in a summary function when using a date or date-time group field.

"daily"
"semimonthly"
"semiannually"
"monthly"
"quarterly"
"biweekly"
"weekly"
"annually"

Time Conditions

The following string values can be used in a summary function when using a time or date-time group field.

"by AMPM"
"by hour"
"by minute"
"by second"

Functions: Type Conversion

This category of functions is used to convert from one data type to another. These functions are modeled after functions of the same name found in Visual Basic and Visual Basic for Applications.

CBool

Returns a Boolean (true/false) conversion of the source field.

CBool will return False for a value of 0 (zero). Otherwise, CBool returns True.

CBool (n)

n – a numeric value.

```
CBool(-1)
```

returns True.

CCur

Returns a currency value.

CCur (v)

v – a value of number, currency, or string data type.

```
CCur ("1125.13894")
```

returns $1,125.14 as a currency value.

CDate

Returns a date value.

CDate is functionally equivalent to Date and DateValue. See **DateValue** for details.

CDateTime

Returns a date-time value.

CDateTime is functionally equivalent to DateTime and DateTimeValue. See **DateTimeValue** for details.

CDbl

Returns a numeric value.

CDbl is functionally equivalent to ToNumber. See **ToNumber** for details.

CStr

Converts any nonstring data type to a string value.

CStr is functionally equivalent to ToText, including returning the same result and accepting the same arguments. See **ToText** for complete information.

CTime

Returns a time value.

CTime is functionally equivalent to Time and TimeValue. See **TimeValue** for details.

Operators: Arithmetic

This category of operators performs standard arithmetic functions, such as addition, subtraction, and so forth.

Add (+)

Adds two numbers.

n1 + n2

n1 – a numeric value.
n2 – a numeric value.

```
2 + 2
```

returns 4.00.

Divide (/)

Divides two numbers.

n1 / n2

n1 – a numeric value.
n2 – a numeric value.

```
10 / 5
```

returns 2.00.

Exponentiate (^)

Raises a number to a power.

n1 ^ n2

n1 – a numeric value.
n2 – a numeric value to raise *n1* to the power of.

```
5 ^ 2
```

returns 25.00.

Integer Divide (\)

Divides two numbers, returning an integer result.

n1 \ n2

n1 – a numeric value.
n2 – a numeric value.

```
10 \ 3
```

returns 3.00, contrasted with:

```
10 / 3
```

which returns 3.33.

Modulus

Performs division on two numeric values and returns the remainder of the division rather than the result of the division.

Use the Mod function for particular situations where the remainder of numeric division is required, rather than the actual result. For example, this operator can be used with the RecordNumber function in a conditional formatting formula to shade every other details section.

n1 Mod n2

n1 – a numeric value
n2 – a numeric value

```
If RecordNumber Mod 2 = 0 Then crSilver Else crNoColor
```

when supplied as a conditional formula for the Details section background color, shades every other details section silver.

NOTE *Crystal Reports also includes the Remainder function, which performs a similar operation.*

Multiply (*)

Multiplies two numbers.

n1 * n2

n1 – a numeric value.
n2 – a numeric value.

```
5 * 4
```

returns 20.00.

Negate (–)

Returns the negative equivalent of a number.

–n

n – a numeric value.

```
-14
```

returns –14.00.

```
-{GLDetails.DRAmount}
```

returns a positive debit amount if the DRAmount field is coded as a negative number in the database.

Percent (%)

Calculates the percentage one number is of another number.

n1 % n2

n1 – a numeric value.
n2 – a numeric value.

```
5 % 20
```

returns 25.00.

```
{Sales.Amount} % Sum({Sales.Amount},{Sales.Rep})
```

returns the percentage each individual sale is of a sales rep's total sales.

Subtract (–)

Subtracts one number from another.

n1 – n2

n1 – a numeric value.
n2 – a numeric value.

```
25 - 10
```

returns 15.00.

Operators: Array

This set of operators pertain to arrays. An *array* is a collection of data items stored in a single "bucket," such as a single variable. If an array contains 15 items, it is said to have 15 *elements.* Arrays can contain any supported Crystal Reports data type, such as number, string, date-time, and so forth.

In

Returns a Boolean (true/false) value indicating whether a single value is in an array.

v In a

v – a value of the same data type as the *a* array.
a – an array.

```
5 In [1,3,5,7,9]
```

returns True.

```
If {Customer.Region} In ["CO","MT","UT","WY"] Then
    "Rocky Mountain Region"
Else
    "Rest of Country"
```

returns one of two strings based on the existence of a database field in the supplied string literal array.

Make Array

Creates an array of values.

This is functionally equivalent to the MakeArray function. See **MakeArray** in the Functions: Arrays section for details.

Redim

"Resets" an existing array variable to an empty state with a specified number of elements.

Redim a (n)

a – an existing array variable.
n – a positive number.

```
WhilePrintingRecords;
StringVar Array Beatles;
Redim Beatles [4];
Beatles[1]
```

declares an existing string array and resets it to a string array with four elements, each containing an empty string. The formula returns an empty string to the report (the first element of the reset array).

Redim Preserve

"Resizes" an existing array variable with a specified number of elements, retaining existing contents of the array variable.

Redim Preserve a (n)

a – an existing array variable.

n – a positive number.

```
WhilePrintingRecords;
StringVar Array Beatles;
Redim Preserve Beatles [8];
Beatles[1]
```

declares an existing string array and expands it to contain eight elements, retaining any existing data in the array. The formula returns "John" to the report (the existing data in the first element of the array).

Subscript ([])

Extracts an individual element of an array.

a[n]

a – an array value.
n – a numeric value or range indicating the element or elements to extract.

```
WhilePrintingRecords;
StringVar Array Beatles;
Beatles[3]
```

returns "George", the third element of the Beatles array.

Operators: Boolean

This set of operators pertain to Boolean (true/false) values, functions, and expressions.

And

Returns true if both associated Boolean values are true.

b1 And b2

b1 – a Boolean value or expression.
b2 – a Boolean value or expression.

```
2 + 2 = 4 And 10 / 2 = 5
```

returns True.

```
If {Customer.Region} = "CA" And
   {Customer.Last Year's Sales} > 50000 Then
    "California Bonus Customer"
Else
    "Other Customer"
```

returns "California Bonus Customer" if both Boolean expressions in the If test are true.

Eqv (Logical equivalence)

Returns true if both associated Boolean values are the same.

b1 Eqv b2

b1 – a Boolean value or expression.
b2 – a Boolean value or expression.

```
2 + 2 = 4 Eqv 10 / 2 = 5
```

returns True.

```
2 + 2 = 8 Eqv 10 / 2 = 5
```

returns False.

```
2 + 2 = 8 Eqv 10 / 2 = 1
```

returns True.

Imp (Logical implication)

Returns true if both associated Boolean values are the same, or if the second value is true while the first value is false.

b1 Imp b2

b1 – a Boolean value or expression.
b2 – a Boolean value or expression.

```
2 + 2 = 4 Imp 10 / 2 = 5
```

returns True.

```
2 + 2 = 8 Imp 10 / 2 = 1
```

returns True.

```
2 + 2 = 4 Imp 10 / 2 = 1
```

returns False.

```
2 + 2 = 8 Imp 10 / 2 = 5
```

returns True.

Not

Reverses the Boolean value (true becomes false and false becomes true).

Not requires a Boolean value or expression to follow it. As such, you may need to enclose a Boolean expression in parentheses for Not to evaluate properly.

Not b

b – a Boolean value or expression.

```
Not (2 + 2 = 4)
```

returns False.

```
Not 2 + 2 = 4
```

results in an error, as Not expects the first occurrence of the number 2 to be Boolean.

```
If Not InRepeatedGroupHeader Then "New Group Starts Here"
```

returns the "new group" string if InRepeatedGroupHeader is false.

Or

Returns true if either or both associated Boolean values are true.

b1 Or b2

b1 – a Boolean value or expression.
b2 – a Boolean value or expression.

```
2 + 2 = 4 Or 10 / 2 = 1
```

returns True.

```
If {Customer.Last Year's Sales} > 50000 Or
   {Customer.Last Year's Sales} < 0 Then
    "Customer needs attention"
Else
    "Normal customer"
```

returns "Customer needs attention" if either Boolean expression in the If test is true.

Xor (Logical exclusion)

Returns true if the associated Boolean values return opposite values (one true, the other false).

b1 Xor b2

b1 – a Boolean value or expression.
b2 – a Boolean value or expression.

```
2 + 2 = 4 Xor 10 / 2 = 6
```

returns True.

```
2 + 2 = 5 Xor 10 / 2 = 5
```

returns True.

```
2 + 2 = 4 Xor 10 / 2 = 5
```

returns False.

Operators: Comparisons

This category of operators compares values to each other. You may combine comparison operators together with other Boolean operators, such as And, Or, and Not.

Equal (=)

Returns a Boolean (true/false) value indicating whether the two supplied values are equal to each other.

v1 = v2

v1 – a value of any supported data type.
v2 – a value of the same data type as *v1.*

```
10 = 10
```

returns True.

```
If {Sales.State} = "CO" Then "Colorado"
```

returns "Colorado" if the state field is equal to "CO".

Greater or Equal (>=)

Returns a Boolean (true/false) value indicating whether the first supplied value is greater than or equal to the second value.

This operator compares strings from the perspective of sort order.

v1 >= v2

v1 – a value of any supported data type.
v2 – a value of the same data type as *v1.*

```
"abc" >= "wyz"
```

returns False, based on string sort order.

```
If {Sales.Amount} >= 5000 Then "Great Order"
```

returns "Great Order" if the sale amount is exactly $5,000, or anything greater than $5,000.

Greater Than (>)

Returns a Boolean (true/false) value indicating whether the first supplied value is greater than the second value.

This operator compares strings from the perspective of sort order.

v1 > v2

v1 – a value of any supported data type.
v2 – a value of the same data type as *v1.*

```
#1/1/2000# > #1/1/1999#
```

returns True.

```
If {Sales.Amount} > 10000 Then "Eligible for Bonus"
```

returns "Eligible for Bonus" if the sale amount is greater than $10,000. If the amount is exactly $10,000 or less, an empty string is returned.

Less or Equal (<=)

Returns a Boolean (true/false) value indicating whether the first supplied value is less than or equal to the second value.

This operator compares strings from the perspective of sort order.

v1 <= v2

v1 – a value of any supported data type.
v2 – a value of the same data type as *v1.*

```
100 <= 100
```

returns True.

```
If {Sales.Amount} <= 100 Then "Small Order"
```

returns "Small Order" if the sale amount is exactly $100, or anything less than $100.

Less Than (<)

Returns a Boolean (true/false) value indicating whether the first supplied value is less than the second value.

This operator compares strings from the perspective of sort order.

v1 < v2

v1 – a value of any supported data type.
v2 – a value of the same data type as *v1.*

```
#1/1/2000# < #1/1/1999#
```

returns False.

```
If {Sales.Amount} < 100 Then "Improved Performance Required"
```

returns "Improved Performance Required" if the sale amount is less than $100. If the amount is exactly $100 or greater, an empty string is returned.

Not Equal (<>)

Returns a Boolean (true/false) value indicating whether the two supplied values are not equal to each other.

v1 <> v2

v1 – a value of any supported data type.

v2 – a value of the same data type as *v1.*

```
10 <> 15
```

returns True.

```
If {Sales.State} <> "CO" Then "Out-Of-State Sale"
```

returns "Out-Of-State Sale" if the state field is something other than "CO".

NOTE *Case sensitivity of string comparisons is based on the database case sensitivity setting in File | Report Options (applies to the current report only) and File | Options (applies to all new reports in the future).*

Operators: Control Structures

This category of operators might well exist in a separate section of the Formula Editor called "programming constructs." However, as they don't take "arguments" per se, they've been placed in the Operator tree. These operators are most familiar to computer programmers, as they duplicate typical programming logic and flow within a single Crystal Reports formula.

Do While

Loops through formula logic while a condition is true.

Do While differs from While Do in regards to when the loop condition is evaluated. Do While evaluates the loop condition *after* a loop iteration (so that at least one loop will always occur). While Do evaluates the condition *before* a loop iteration (the logic within the loop may not occur at all if the condition is immediately false).

Do <formula logic> While b

b – a Boolean value or expression.

```
NumberVar Counter := 1;
StringVar Accum;
do
(   Accum := Accum + "-";
    Counter := Counter + 1)
while Counter < 100;
Accum
```

returns a string containing 99 dash characters (the loop iterated while the counter was less than 100—once it reached 100, the loop did not repeat).

NOTE *If you wish to include more than one Crystal Reports statement within the loop, separate statements with a semicolon and surround all statements (between the Do and While) with parentheses.*

Exit For

Exits a For loop before its normal conclusion.

Exit For

```
NumberVar Counter;
StringVar Accum;
for Counter := 1 to 1000 step 2 do
(
    Accum := Accum & ToText(Counter);
    If Len(Accum) > 245 Then Exit For;
);
Accum
```

returns the string in the Accum variable consisting of "1.003.005.007.009.00" and so forth. Within the loop, the Accum variable is tested for a length exceeding 245 characters and the loop is exited if this occurs.

Exit While

Exits a Do or While loop before its normal conclusion.

Exit While

```
NumberVar Counter := 1;
StringVar Accum;
do
(   Accum := Accum + "-";
    Counter := Counter + 1;
    If Length(Accum) = 100 Then Exit While)
while True;
Accum
```

returns a string containing 100 dash characters (the loop iterates until the length of the Accum variable reaches 100 characters and the Exit While statement executes).

For

Loops through formula logic a specified number of times.

For v := n1 To n2 Step n3 Do

v – a numeric variable that the loop will increment as it progresses.
n1 – a numeric value indicating the beginning value that will be assigned to *v* when the loop starts.
n2 – a numeric value indicating the ending value in *v* that will stop the loop.
n3 – a numeric value indicating the amount to increment *v* every time the loop iterates. This value, as well as the Step keyword that precedes it, are optional. If omitted, *v* is incremented by 1 with each loop iteration.

```
NumberVar Counter;
StringVar Accum;
```

```
for Counter := 1 to 100 step 2 do
(
    Accum := Accum & ToText(Counter);
);
Accum
```

returns contents of the Accum variable, consisting of the string "13579111315" and so forth, through "99". The Counter variable increments from 1 to 100, incrementing by 2 every time the loop iterates.

NOTE *If you wish to include more than one Crystal Reports statement within the loop, separate statements with a semicolon and surround all statements (after the Do keyword) with parentheses.*

If Then Else

Performs a Boolean test, returning one value if the test is true and an alternate value if the test is false.

If b Then v1 Else v2

b – a Boolean expression (typically using comparison operators discussed earlier in this section).
v1 – a value of any supported data type that is returned if *b* evaluates to True.
v2 – a value of the same data type as *v1* that is returned if *b* evaluates to False. This value, and the Else keyword that precedes it, is optional.

```
If {Sales.State} = "CO" Then
   "Sales Tax Required"
Else
   "No Tax"
```

returns the string "Sales Tax Required" if the Boolean expression after If evaluates to true. Otherwise, the formula returns "No Tax".

```
If {Sales.Amount} > 5000 Then "Bonus Order"
```

returns "Bonus Order" if the sales amount exceeds $5,000. Otherwise, an empty string is returned.

Option Loop

Specifies how many iterations a loop should go through before an error is returned.

By default, Crystal Reports stops loop processing and returns an error if a loop iterates 100,000 times. This behavior prevents infinite loops from occurring. If you wish to lower the maximum number of iterations, use Option Loop.

Option Loop n

n – a numeric value indicating the number of loop iterations that will occur before an error is returned.

Option Loop must be the first statement in the formula.

```
NumberVar Counter;
Do
    Counter := Counter + 1
While True
```

will iterate 100,000 time before returning an error.

```
Option Loop 100;
NumberVar Counter;
Do
    Counter := Counter + 1
While True
```

will iterate 100 times before returning an error.

Select Case

Returns one of several available values based on a series of conditions.

Select Case duplicates the capabilities of sophisticated If-Then-Else logic. However, Select Case is often easier to understand and maintain than complex If-Then-Else formulas.

Select e Case v1: <formula logic> Case v2 <formula logic>...: Default: <formula logic>

e – a value or expression of any data type to test.
v1 – a value or "list" of values (separated by commas) matching in data type to *e* to test against *e*.
v2 – an additional value or "list" of values (separated by commas) matching in data type to *e* to test against *e*.

Additional pairs of Case/<formula logic> combinations may be added. The Default keyword and colon are optional. Formula logic after each Case statement must return the same data type as all other formula logic.

```
Select {Sales.State}
    Case "OR","ID","MT","WA":
        "Northwest"
    Case "CA","AZ","TX","NM":
        "Southwest"
    Case "ME","MA","NH","NY":
        "Northeast"
    Case "FL","NC","GA","SC":
        "Southeast"
    Default:
        "Rest of Country"
```

examines the state field and begins testing it against each Case statement. Once it finds a match, it returns the value following the case statement. If no matches are found, the value following the default statement is returned. Were the Default keyword, colon, and "Rest of Country" string literal to be left out, the formula would return an empty string if no matches were found.

```
Select {Sales.LastYearRevenue}
    Case Is < 100:
        "Improved Performance Needed"
    Case 100 To 1000:
        "Average Year"
    Case Is > 1000:
        "Excellent Year"
```

examines the Last Year Revenue field and begins testing against the ranges specified in each Case statement, returning the appropriate string following the Case statement that matches. No Default keyword is provided, as any number will fall into one of the three Case statements. Were a number to somehow not fall into one of the existing Case statements, a zero would be returned.

While Do

Loops through formula logic while a condition is true.

While Do differs from Do While in regards to when the loop condition is evaluated. Do While evaluates the loop condition *after* a loop iteration (so that at least one loop will always occur). While Do evaluates the condition *before* a loop iteration (the logic within the loop may not occur at all if the condition is immediately false).

While b <formula logic> Do

b – a Boolean value or expression.

```
NumberVar Counter := 1;
StringVar Accum;
While Counter < 100 Do
(   Accum := Accum + "-";
    Counter := Counter + 1
);
Accum
```

returns a string containing 99 dash characters (the loop iterated while the counter was less than 100—once it reached 100, the loop did not repeat).

NOTE *If you wish to include more than one Crystal Reports statement within the loop, separate statements with a semicolon and surround all statements (between the Do and While) with parentheses.*

Operators: Conversion

A single operator exists to convert to one data type from another.

Currency ($)

Converts a numeric value to a currency value.

The CCur function can also be used to convert from another data type to currency.

$

```
$100
```

returns $100.00.

```
${Sales.Amount} * 1.1
```

returns 10 percent above the sales amount as a currency value.

NOTE *Crystal Reports prohibits both values in a multiplication formula from being currency—one or the other can be, but not both. If necessary, you'll need to convert one currency value to a numeric value with ToNumber or CDbl.*

Operators: Other

This category of operators contains, in essence, operators that don't fit in any other category and perform miscellaneous functions.

Assignment (:=)

Assigns a value to a variable.

In Crystal syntax, don't confuse the assignment operator (:=) with the equals comparison operator (=). In Basic syntax, equals (=) acts as both assignment and equals comparison.

var := v

var – a previously declared variable name.
v – a value or expression of the same data type as *var*.

```
SalesRepBonus := SalesRepBonus + 1
```

adds 1 to the value already in the SalesRepBonus variable and places the result back into the SalesRepBonus variable.

```
NumberVar SalesRepBonus := 0
```

declares a number variable named SalesRepBonus and assigns it a value of zero in a single statement.

Comment (//)

Treats any text following the two slashes as a comment.

Basic Syntax uses the apostrophe (') or the word Rem to indicate a comment. Crystal Reports allows you to add comment slashes or apostrophes to multiple formula lines at once by highlighting them and clicking the Comment/Uncomment button in the Formula Workshop toolbar.

```
//
//The following formula calculates
//commission based on sale amount
Select {Orders.Order Amount}
```

```
Case Is < 100:
    .01
Case 100 to 1000:
    .05
Case Is > 1000:
    .1
```

The first two lines of the formula are ignored by the Formula Editor when evaluating the formula logic.

NOTE *If you are troubleshooting a formula, you may prefer to simply "comment out" certain formula lines that you wish to later use again. For example, you may be working on five different methods of calculating a result. By adding or removing two slashes in front of the various formula lines, you may try various methods of calculations without having to delete and retype each line.*

Date-time literal (#)

Returns a date-time value from the supplied string.

#s#

s – a string that can be interpreted as a date, time, or date-time combination.

```
#1/1/2000 10:15 am#
```

returns a date-time value of 1/1/2000 10:15:00AM.

```
#Sep 10, 02#
```

returns a date-time value of 9/10/2002 12:00:00AM.

NOTE *Crystal Reports is fairly creative in what it will interpret as date or time material between the # characters. If the string can't be understood, an error will occur.*

Parentheses

Used to force evaluation of formula expressions in a certain order.

Parentheses allow you to force calculations to occur in other than the default *order of precedence* (exponentiation, then multiplication/division left to right, then addition/subtraction left to right). In Basic syntax, parentheses are also used to delimit array subscripts.

(<expression>)

```
10 + 10 * 2
```

returns 30.00. The order of precedence causes the multiplication to be done first, then the addition.

```
(10 + 10) * 2
```

returns 40.00. The parentheses force the addition to be performed first, then the multiplication.

Operators: Pattern

This set of operators allow partial string matches to be evaluated.

Like

Returns a Boolean (true/false) value based on a partial string match.

Like uses DOS-style "wildcards" (the asterisk and question mark) to determine if the source string contains the characters specified in the wildcard-based mask. This is helpful for checking for partial text matches. This operator is similar to the LooksLike function.

s1 Like s2

s1 – the source string to be searched.
s2 – the mask string to search against. The mask can include a question mark wildcard to indicate a single-character substitution and/or an asterisk to indicate a multiple-character substitution.

```
"George Peck" Like "G?orge*"
```

returns True.

StartsWith

Returns a Boolean (true/false) value if leading characters match the source string.

s1 StartsWith s2

s1 – the source string to be searched.
s2 – the characters to search for as the first characters of *s1.*

```
"George Peck" StartsWith "Ge"
```

returns True.

NOTE *Case sensitivity of string comparisons is based on the database case sensitivity setting in File | Report Options (applies to the current report only) and File | Options (applies to all new reports in the future).*

Operators: Ranges

These operators either create various ranges of values or test existing ranges. A *range* of values consists of beginning and ending values, and every value in between (some ranges can have no beginning and/or ending value—in other words, the range can be "everything above x" or "everything below y").

Both Endpoints Excluded Range

Creates a range of values between the two endpoints, not including the endpoints.

v1 _To_ v2

v1 – a value of any data type to act as the lower endpoint.

v2 – a value of the same data type as *v1* to act as the upper endpoint.

```
1 In (1 _to_ 100)
```

returns False, as the lower endpoint (1) is not included in the range.

```
99 In (1 _to_ 100)
```

returns True.

In Range

Returns a Boolean (true/false) value indicating whether the single value is in the range.

v In r

v – a single value of any supported data type.
r – a range value or variable of the same data type as *v*.

```
5 In (1 To 100)
```

returns True.

```
If {Parts.PartNo} In (1000 to 5000) Then
   "Taxable"
Else
   "Non-taxable"
```

returns the string "Taxable" if the part number is in the range of 1000 to 5000 inclusive.

Left Endpoint Excluded Range

Creates a range of values between the two endpoints, not including the first endpoint.

v1 _To v2

v1 – a value of any data type to act as the lower endpoint.
v2 – a value of the same data type as *v1* to act as the upper endpoint.

```
1 In (1 _to 100)
```

returns False, as the lower endpoint (1) is not included in the range.

```
100 In (1 _to 100)
```

returns True, as the upper endpoint (100) is included in the range.

Make Range

Creates a range of values between, and including, the two endpoints.

v1 To v2

v1 – a value of any data type to act as the lower endpoint.

v2 – a value of the same data type as *v1* to act as the upper endpoint.

```
1 In (1 to 100)
```

returns True.

```
100 In (1 to 100)
```

returns True.

Right Endpoint Excluded Range

Creates a range of values between the two endpoints, not including the second endpoint.

v1 To_ v2

v1 – a value of any data type to act as the lower endpoint.
v2 – a value of the same data type as *v1* to act as the upper endpoint.

```
1 In (1 to_ 100)
```

returns True, as the lower endpoint (1) is included in the range.

```
100 In (1 to_ 100)
```

returns False, as the upper endpoint (100) is not included in the range.

UpFrom

Creates a range of values including a lower endpoint upward, with no upper endpoint.
Is >= v is functionally equivalent to UpFrom.

upFrom v

v – a value of any data type to act as the lower endpoint.

```
99999999 In UpFrom 100
```

returns True.

```
HasUpperBound (UpFrom 100)
```

returns False (the HasUpperBound function evaluates whether the supplied range has an upper endpoint).

Up From But Not Including

Creates a range of values from a lower endpoint upward (the lower endpoint not being included), with no upper endpoint.
Is > v is functionally equivalent to UpFrom_.

upFrom_ v

v – a value of any data type to act as the lower endpoint.

```
100 In UpFrom_ 100
```

returns False (the lower endpoint is not included in the range).

```
HasUpperBound (UpFrom_ 100)
```

returns False (the HasUpperBound function evaluates whether the supplied range has an upper endpoint).

UpTo

Creates a range of values from an upper endpoint downward (including the upper endpoint) with no lower endpoint.

Is <= v is functionally equivalent to UpTo.

upTo v

v – a value of any data type to act as the lower endpoint.

```
-9999999 In UpTo 100
```

returns True.

```
HasLowerBound (UpTo 100)
```

returns False (the HasLowerBound function evaluates whether the supplied range has a lower endpoint).

Up To But Not Including

Creates a range of values from an upper endpoint downward (not including the upper endpoint) with no lower endpoint.

Is < v is functionally equivalent to UpTo_.

upTo_ v

v – a value of any data type to act as the lower endpoint.

```
100 In UpTo_ 100
```

returns False (the upper endpoint is not included in the range).

```
HasLowerBound (UpTo_ 100)
```

returns False (the HasLowerBound function evaluates whether the supplied range has a lower endpoint).

Operators: Scope

This set of operators is used when declaring variables (see "Operators: Variable Declaration" later in this appendix). These operators determine how long a variable retains its value during report processing.

Global

Forces a variable to retain its value in this formula, and all other formulas in the current report (but not subreports).

In Crystal Syntax, this is the default scope for variable declarations that do not include a scope keyword.

Global t v

t – a variable declaration statement.
v – a valid variable name.

```
Global NumberVar BonusCount
```

declares a numeric variable called BonusCount that will retain its value throughout the entire report, but not in any subreports.

```
NumberVar BonusCount
```

is functionally equivalent to the previous declaration, as Global is the default variable scope.

Local

Forces a variable to retain its value in this formula only.

In Basic Syntax, this is the default scope for variable declarations that do not include a scope keyword.

Local t v

t – a variable declaration statement.
v – a valid variable name.

```
Local NumberVar BonusCount
```

declares a numeric variable called BonusCount that will retain its value only during the calculation of this formula. If BonusCount is declared in any other formulas, it will contain zero.

Shared

Forces a variable to retain its value in this formula, in all other formulas in the current report, and in all subreports.

Shared t v

t – a variable declaration statement.
v – a valid variable name.

```
Shared NumberVar BonusCount
```

declares a numeric variable called BonusCount that will retain its value throughout the entire report and in all subreports.

Operators: Strings

This set of operators applies to string manipulation, such as string "concatenation" (the process of combining two or more strings into a single string).

Concatenate (& or +)

Concatenates (combines) strings into a single string.

The + operator requires that surrounding values be strings, while the & operator performs an implicit conversion to string of all surrounding values.

s1 + s2

s1 – a string value.
s2 – a string value.

```
"Page Number: " + PageNumber
```

returns an error, as the PageNumber function returns a numeric value.

```
"Page Number: " + ToText(PageNumber,0)
```

returns "Page Number: 1" if the first page of the report is printing.

v1 & v2

v1 – a value of any data type.
v2 – a value of any data type.

```
"Page Number: " & PageNumber
```

returns "Page Number: 1.00" if the first page of the report is printing, performing an implicit conversion of PageNumber to a string data type.

```
"Page Number: " & ToText(PageNumber,0)
```

returns "Page Number: 1" if the first page of the report is printing.

NOTE *When using the & operator to concatenate strings, you may still need to use ToText or CStr to control how values are formatted when they are concatenated.*

In String

Returns a Boolean value (true/false) based on if the first string is contained in the second string.

s1 In s2

s1 – a string value to search for in *s2.*

s2 – the source string to search.

```
"eor" in "George"
```

returns True.

```
If "(303)" In {Customer.Phone} Then "Denver Area Code"
```

returns the string "Denver Area Code" if the characters "(303)" are contained anywhere in the customer phone database field. Otherwise, the formula returns an empty string.

Insert Empty String ("")

Inserts a pair of quotation marks in the formula.

In Crystal Reports versions prior to 9 (as well as in 9), you may simply type quotation mark pairs in directly.

""

```
StringVar BonusCustomer := ""
```

declares a string variable called BonusCustomer and sets it to contain an empty string.

Subscript []

Extracts a substring from a larger string.

s[n]

s – a string value.
n – a numeric value or range indicating the character or characters to extract.

```
"George Peck"[2 to 4]
```

returns "eor".

Operators: Variable Declarations

This category of operators is used to declare variables within formulas. Whenever a variable is used in any formula, it must first be declared using one of the following operators.

BooleanVar

Declares a Boolean (true/false) variable to contain an array or single value.

BooleanVar Array varname

BooleanVar varname

varname – a variable name that is not the same as any other Crystal Reports formula keyword, does not contain a space, and does not start with a number or certain special characters.

```
BooleanVar Array Workdays :=
   [False, True, True, True, True, True, False];
Workdays[DayOfWeek(CurrentDate)]
```

returns True if the CurrentDate is Monday through Friday. The formula declares a Boolean array variable containing seven elements and then extracts the element associated with the day of the week.

```
BooleanVar BonusReached;
If {Sales.Amount} > 5000 Then BonusReached := True
```

declares a Boolean variable and assigns it a value of True if a sales amount is exceeded.

CurrencyVar

Declares a currency variable to contain an array, array of ranges, range, or single value.

CurrencyVar Array varname

CurrencyVar Range Array varname

CurrencyVar Range varname

CurrencyVar varname

varname – a variable name that is not the same as any other Crystal Reports formula keyword, does not contain a space, and does not start with a number or certain special characters.

```
CurrencyVar Range GoodSales := upFrom 5000;
If {Sales.Amount} In GoodSales Then "Good Job"
```

declares a currency range variable and assigns it all values including and above $5,000. If a sale amount is included in the variable, "Good Job" is returned by the formula.

```
CurrencyVar HighAmount;
If {Sales.Amount} > HighAmount Then
    HighAmount := {Sales.Amount}
```

declares a currency variable. If the sales amount is higher than what's retained in the variable from previous records, the variable is assigned the value of the higher sales amount.

DateTimeVar

Declares a date-time variable to contain an array, array of ranges, range, or single value.

DateTimeVar Array varname

DateTimeVar Range Array varname

DateTimeVar Range varname

DateTimeVar varname

varname – a variable name that is not the same as any other Crystal Reports formula keyword, does not contain a space, and does not start with a number or certain special characters.

```
DateTimeVar Range WorkDays :=
     #6/2/2003 8:00am# To #6/6/2003 5:00pm#;
If Not ({Salary.WorkDate} In WorkDays) Then
    {Salary.DailyPay} + {Salary.OvertimePay}
Else
    {Salary.DailyPay}
```

declares a date-time range variable and assigns it a value of Monday at 8 A.M. through Friday at 5 P.M. The variable is then checked against a work date to determine if overtime should be added to an employee's pay.

```
DateTimeVar OutOfTolerance;
If {Meas.Sample Value} > {Standards.Sample} Then
    OutOfTolerance := {Meas.Sample Date Time}
```

declares a date-time variable and tests to see if a sample reading was out of tolerance. If so, the date-time that the exception occurred is assigned to the variable.

DateVar

Declares a date variable to contain an array, array of ranges, range, or single value.

DateVar Array varname

DateVar Range Array varname

DateVar Range varname

DateVar varname

varname – a variable name that is not the same as any other Crystal Reports formula keyword, does not contain a space, and does not start with a number or certain special characters.

```
DateVar Range Array CompanyHolidays :=
   [DateValue("1/1/2003") to DateValue("1/2/2003"),
    DateValue("2/17/2003"),DateValue("5/26/2003"),
    DateValue("7/4/2003"),DateValue("9/1/2003"),
    DateValue("11/27/2003") to DateValue("11/28/2003"),
    DateValue("12/25/2003") to DateValue("12/31/2003")];
If {Salary.WorkDate} In CompanyHolidays Then
    "Bonus Pay Required"
```

declares a date range array and sets it to the individual dates and date ranges that make up company holidays. The variable is then checked to see if a work day qualifies for bonus pay.

NumberVar

Declares a number variable to contain an array, array of ranges, range, or single value.

NumberVar Array varname

NumberVar Range Array varname

NumberVar Range varname

NumberVar varname

varname – a variable name that is not the same as any other Crystal Reports formula keyword, does not contain a space, and does not start with a number or certain special characters.

```
NumberVar SalesRepTotal;
If {Sales.Amount} > {SalesRep.BonusLevel} Then
    SalesRepTotal :=
       SalesRepTotal + {Sales.Amount}
```

declares a number variable and tests to see if a sales amount exceeded a bonus level. If so, the variable is incremented by the amount of the sale.

```
NumberVar SalesRepTotal := 0
```

declares a number variable and resets it to zero in the same formula statement.

StringVar

Declares a string variable to contain an array, array of ranges, range, or single value.

StringVar Array varname

StringVar Range Array varname

StringVar Range varname

StringVar varname

varname – a variable name that is not the same as any other Crystal Reports formula keyword, does not contain a space, and does not start with a number or certain special characters.

```
StringVar ShippersUsed;
If Not ({Orders.Ship Via} In ShippersUsed) Then
    ShippersUsed := ShippersUsed & {Orders.Ship Via} & ", "
```

declares a string variable. A test is performed to see if the shipper database field is already contained in the variable. If not, the shipper database field is added to what's already in the variable, concatenated with a comma and space.

```
StringVar Shippers;
Left(Shippers, Length(Shippers)-2)
```

declares a string variable and returns all but the right two characters of the variable to the report.

TimeVar

Declares a time variable to contain an array, array of ranges, range, or single value.

TimeVar Array varname

TimeVar Range Array varname

TimeVar Range varname

TimeVar varname

varname – a variable name that is not the same as any other Crystal Reports formula keyword, does not contain a space, and does not start with a number or certain special characters.

```
TimeVar Range WorkHours;
If DayOfWeek({Sales.Date}) In 2 to 6 Then
    WorkHours := TimeValue("9:00 am") To TimeValue("7:00 pm")
Else
    WorkHours := TimeValue("10:00 am") To TimeValue("5:00 pm");
"Store Hours: " & Minimum(WorkHours) &
              " to " & Maximum(WorkHours)
```

declares a time range variable. The day of the week is checked and the variable is given an hour range for weekdays or weekends. The formula returns a string extracting the lower bound and upper bound of the range.

NOTE *Some of the previous examples illustrate the capability of declaring a variable and assigning it a value in the same formula statement.*