crystal decisions™

Crystal Reports

Additional Functions you can add to Crystal Reports

Overview

This document is intended for report designers and developers who wish to add additional functions to the Crystal Reports Formula Editor. It lists and describes all the supported Additional User Functions contained in various UFL files available on the Crystal Reports CD and/or website, and provides instructions for locating and installing these additional functions.

Applies to version 4 or higher of Crystal Reports, and to version 5 or higher of the Seagate Info Report Designer.

To navigate this document more easily, display the document map in Word: on the View menu, click Document Map.

This displays a collapsible tree at left, which you can use to jump directly to a section within the document.

Contents

Introduction - what are UFLs?

The formula language of Crystal Reports is expandable. The Formula Editor has the ability to accept new functions created by developers for specific needs. We refer to these new functions as *additional user function libraries* (UFL).

Additional UFLs are not included when you install Crystal Reports. You need to install them separately. Once you have installed the necessary UFL file you will see the new function(s) in the Additional Functions folder of the Crystal Reports Formula Editor.

UFLs abide by the following naming conventions:

- Files starting with "UFL" are for use with any 16-bit version of Crystal Reports
- Files starting with "U2L" are for use with any 32-bit version of Crystal Reports
- Files starting with "UF5" are for use with 16-bit 5.0.x.108 and higher
- Files starting with "U25" are for use with 32-bit 5.0.x.108 and higher

NOTE	Although there are four different naming conventions used by User Function Libraries, throughout this document they will all be referred to as "UFLs".

A single UFL file (Ufl*name*.dll) may contain more than one function. Once installed, a UFL's additional functions will display automatically in the formula editor the next time you start Crystal Reports, just like regular functions.

Using this document

- If you know the name of the function, but don't know the name of the EXE file containing it, go to the Appendix on page 26.
- If you don't know the name of a function, or you want to check if a particular kind of additional function exists, go to *Function listing by function type* on page 5.
- If you know the name of the EXE, to find out where to obtain it, go to *Obtaining UFL files* on page 3.
- To learn how to install the UFL, you can either read the Readme.txt or Install.txt file included with the EXE, or you can go to *Adding an additional function* on page 4.

Where do you obtain UFL files?

There are two places where you can obtain UFLs on your own:

- From the Crystal Decisions support website.
- From the Crystal Reports CD.

The additional UFLs are packaged in self-extracting EXEs. The appendix provides an alphabetical index of additional functions, indicating the name of the self-extracting EXE that contains the function.

To download the EXE from our web site

1. Go to http://support.crystaldecisions.com/downloads.
2. In the Search by filename box, type the EXE filename.
3. Press Enter.

You can find additional UFLs on the Crystal Reports CD in the UFL folder. The self-extracting EXEs containing the UFL files are stored in subfolders of the same name.

To obtain a UFL from the Crystal Reports CD

1. Use Windows Explorer to navigate to the Crystal Reports 6 or 7 CD.
2. Open the UFL folder.
3. Locate the appropriate sub-folder then run the EXE to extract the contents to your hard drive.

See the section below for installation instructions

Working with UFLs

The following instructions apply to most of the UFLs listed. The two exceptions are UflMath.dll & UflConv.dll. Instructions for installing these UFLs are included with their descriptions in this document.

You can also find installation instructions in the Readme.txt or Install.txt file included with the EXE.

To add an additional function

1. Consult this document for the location of UFL file.
2. Download the UFL file. The self-extracting file will contain an INSTALL.TXT, the DLL file(s), and if available, an UFLcode.EXE containing the source code used to create the DLL(s).
3. Move the DLL file(s) to:
   - In Windows 9x:

     \WINDOWS\CRYSTAL (for developer machines)

     \WINDOWS\SYSTEM (for client machines).
   - In Windows NT:

     \WINNT\CRYSTAL (for developer machines)

     \WINNT\SYSTEM (for client machines).
4. Restart Crystal Reports and the functions will be automatically detected and added to the bottom of the middle pane in the formula editor (in the “Additional Functions” area).

Use the functions within the formula editor just like any other formula.

Functions listed by type

These descriptions are based on the readme.txt or install.txt files included along with the UFL files in the self-extracting EXE. They include:

- The EXE filename
- The versions of Crystal Reports the UFL is compatible with
- The names of the functions this UFL adds
- Description of the function's purpose
- The 32-bit and 16-bit filenames for the DLL file
- Formula Syntax and examples

To enable you to 'browse' the available functions, this section is subdivided by type of function: Date or Time, String, Arithmetic, and other.

Date or Time functions

This sections describes additional functions to

- Create a 21st century date
- Determine day of week or day of year
- Calculate the difference between two date times
- Convert a Paradox Time field to a Crystal Reports time format
- Determine the number of seconds since midnight
- Convert a number to a Crystal Reports date format.

21st Century Date UFL

Filename	UF52000.EXE ** For use with Crystal Reports v5 and higher. **
Description	This function will convert Date field, Date-Time field, or Date-Time String "x" to a date in the 21st century IF the last 2 digits of the year are less than a user-specified number "y".
32-bit DLL	U2L2000.DLL U252000.DLL
16-bit DLL	UFL2000.DLL UF52000.DLL

Adds functions	DateTo2000(Date, Number)
	DateTimeTo2000(DateTime, Number)
	DTSTo2000(DateTimeString, Number)

Formula Syntax

```
DateTo2000(x,y)
```

This function will convert a date field "x" to a date in the 21st century if the last 2 digits of the year are less than the number "y".

`DateTimeTo2000(x,y)`

This function will convert a DATE/TIME field "x" to a date in the 21st century IF the last 2 digits of the year are LESS than the number "y".

`DTSTo2000(x,y)`

This function will convert a DATE/TIME STRING field "x" to a date in the 21st century IF the last 2 digits of the year are LESS than the number "y".

Examples

{DateField} is a Date field in the database with a 2-digit year of 48. {DateTime} is a Date/Time field in the database with a 2-digit year of 47. {DTS} is a Date/Time String field in the database with a 2-digit year of 46.

`DateTo2000({DateField}, 50)`	Returns Date(2048,MM,DD)
`DateTimeTo2000({DateTime}, 50)`	Returns Date(2047,MM,DD)
`DTSTo2000({DTS}, 50)`	Returns Date(2046,MM,DD)

In these examples, any dates in the fields where the last two digits of the year are less than or equal to 50, will be changed to 21st century dates. Dates in the fields that are GREATER than 50 will stay as 20th century dates. If you want to have ALL dates converted to 21st century dates, put the number 99 in the "y" position of the function.

Btrieve Time UFL

Filename	UFLBTIME.EXE Compatible with version 4.5 or higher
Description	Used to convert number fields that represent a fraction of a day (24 hours) into a time string in the format "HH:MM:SS".
32-bit DLL	U2LBTIME.DLL
16-bit DLL	UFLBTIME.DLL
Adds functions	BTime(x)

Formula Syntax

`BTime(x)`	Where "x" is a decimal number representing a fraction of a day.

Examples

Where {numberfield} is a NUMERIC decimal field representing a fraction of a day between 0 and 1.

`BTime(.25)`	Returns the string "06:00:00"
`BTime(.50)`	Returns the string "12:00:00"
`BTime(.66)`	Returns the string "15:50:24"
`BTime(.75)`	Returns the string "18:00:00"
`BTime(1.0)`	Returns the string "24:00:00"

Day of Year, Week of Year

Filename	UFLDATE.EXE Compatible with versions: 4.5 and higher
Description	Adds two functions to the "Additional Functions" portion of the Formula Editor. DayOfYear returns a numeric value of the day of the year that a Date field represents. WeekOfYear returns a numeric value of the number of the week that a Date field falls in for the given year.
32-bit DLL	U2LDATE.DLL
16-bit DLL	UFLDATE.DLL
Adds functions	DayOfYear — Returns a number representing the day of the year (1-366) WeekOfYear — Returns a number representing the week number (1-53) based on Sunday being the first day of a week.

Formula Syntax

```
DayOfYear({datefield})

WeekOfYear({datefield})
```

Examples

`DayOfYear(Date(1998,01,25))`	Returns 25.00
`WeekOfYear(Date(1998,01,25))`	Returns 5.00 (Sunday is 1st day of week)

Date Time Difference UFL

Filename	UFLDTDIF.EXE Compatible with versions: 5.0 and higher
Description	This UFL adds the ability to calculate the difference between two date/time fields, and returns it as: "xx days xx hours xx minutes xx seconds"
32-bit DLL	U2LDTDIF.DLL
16-bit DLL	UFLDTDIF.DLL
Adds functions	DateTimeDiff (datetime, datetime)

Formula Syntax

`DateTimeDiff (x, y)` Where "x" and "y" are true date/time fields, not d date/time strings.

Examples

- {datetime1} is equal to: January 15, 1997 12:15:00 PM
- {datetime2} is equal to: January 16, 1997 1:30:00 PM

```
DateTimeDiff ({datetime1}, {datetime2})
```

Returns the STRING:

```
"1 day(s)  1 hour(s), 15 minute(s), 0 second(s)"
```

Date Add UFL

Filename	UFLDATEADD.EXE Compatible with version 6 and 7
Description	This User Function Library (UFL) implements the DateAdd function. The function adds a specified number of intervals (month, day, year, and so on) to a date. This function is installed automatically with CR version 8.
32-bit DLL	U25DateAdd.dll
16-bit DLL	N/a
Adds functions	DateAdd(interval, number, date)

Formula Syntax

```
DateAdd(interval, number, date)
```

Arguments	Description
Interval	Required. String expression that is the interval of time you want to add.
Number	Required. Numeric expression that is the number of intervals you want to add. It can be positive (to get dates in the future) or negative (to get dates in the past).
Date	Required. Variant (Date) or literal representing date to which the interval is added.

Interval Setting	Description
yyyy	Year
q	Quarter
m	Month
Y	Day of year
d	Day
w	Weekday (same as Day)
ww	Week
h	Hour
n	Minute
s	Second

Formula Syntax

```
DateAdd ("m",1 ,DateTime(2000,1,31,1,2,3))
```

Returns:

```
Feb 29, 2000 1:02:03 AM
```

Julian to Date UFL

For more information on Julian Dates, refer to knowledge base article C2009191.

Filename	UFLJUL.EXE Compatible with versions: 4.0 and higher
Description	Adds new functions to allow conversion to and from Julian Date fields.
32-bit DLL	U2LJUL.DLL
16-bit DLL	UFLJUL.DLL

Adds functions	DateToJulian(x)
	JulianToDate(x)

Formula Syntax

`DateToJulian (date)` Takes a date field and returns the Julian date value.

`JulianToDate (number)` Takes a Julian date and returns the date value.

Examples

`DateToJulian (date(1995,12,25))`

Would convert 12/25/1995 to 2450077 (the Julian date)

`JulianToDate ({table.datefield})`

Would convert 2450077 (the Julian date) to a 12/25/1995.

Some date fields use a special julian-type date format that is based on a particular date, for example January 1, 1900. The following formulas would convert them:

To convert the date 12/25/1995 to the special Julian date:

```
DateToJulian(<date>) - DateToJulian(Date(1900,1,1)-1)

//1 is subtracted from the special date 'start time'
//so the start date is included when calculating the
//difference.
```

To convert a special Julian date field to normal date:

```
JulianToDate( DateToJulian(Date(1900,01,01)-1) +
{table.date} )
```

Paradox Time UFL

Filename	UFLPDXTM.EXE Compatible with versions: 5.0 and higher
Description	Converts Paradox Time fields to Crystal Reports Time format
32-bit DLL	U2LPDXTM.DLL
16-bit DLL	UFLPDXTM.dll
Adds functions	PDXTimeToCRTime(number)

Paradox Time fields are read by Crystal Reports as numbers, representing Seconds after Midnight. This is the way Paradox stores it's time information. It is possible to take this integer value and convert it into a formatted string of:

HH:MM:SS AM/PM by creating a formula in the Crystal Reports formula editor.

However, this new UFL does the conversion automatically, and converts the field into Crystal Reports Time format, which allows more flexibility in formatting, and includes support for the Microsoft Windows Time-format settings.

Formula Syntax

`PDXTimeToCRTime({table.timefield})`

Examples

`PDXTimeToCRTime({table.timefield})`

Returns the time in HH:MM:SS AM/PM format (Crystal Reports Time format)

If you are going to do calculations with the Paradox Time fields (such as Time minus Time), you will probably want to do those calculations with the original fields since it is easier to do the calculations with numeric Seconds. When you are done your calculations and are ready to display the result in formatted fashion, just place the result in the PDXTimeToCRTime() function.

NOTE	Attempting to pass the PDXTimeToCRTime() function a number that is out of range (less than zero, or greater than 86399) will generate an error in the Formula Editor.

Seconds Since Midnight UFL

Filename	UFLSSM.EXE Compatible with versions: 5.0 and higher
Description	This UFL adds functions to convert Time fields that are stored as Seconds since Midnight to standard output formats.
32-bit DLL	U2LSSM.DLL
16-bit DLL	UFLSSM.DLL
Adds functions	HoursFromSecsSinceMidnight ({Table.Time}) - returns the hours since midnight as an number. MinutesFromSecsSinceMidnight ({Table.Time}) - returns the minutes since midnight as an number. SecondsFromSecsSinceMidnight ({Table.Time}) - returns the seconds since midnight as an number. HHMMPMFromSecsSinceMidnight({Table.Time}) - returns the time as a string in the format HH:MM PM HHMMSSPMFromSecsSinceMidnight({Table.Time}) - returns the time as a string in the format HH:MM:SS PM HHMMFromSecsSinceMidnight({Table.Time}) - returns the time as a string in a military time format HH:MM

HHMMSSFromSecsSinceMidnight({Table.Time})
- returns the time as a string in a military time format

HH:MM:SS

CRTimeFromSecsSinceMidnight({Table.Time})
- returns a Crystal Reports Time field for 5.0

WeekOfYear({Table.Datefield})
- Returns the Week of the Year as a Number, much like the Day Of Week Function

NOTE	These UFLs could be made compatible with Crystal Reports 4.x if the function "CRTimeFromSecsSinceMidnight({Table.Time})" was left out, since it uses a Crystal Reports Time type that was not available with Crystal Reports 4.x versions.

Number to Date UFL

Filename	UFLTDATE.EXE Compatible with versions: 4.0 and higher
Description	This adds a function allowing you to convert numeric fields in the format YYMMDD or YYYYMMDD into the Crystal Reports date format Date(YYYY,MM,DD). You can then modify the Crystal Reports date format to control how the date displays on the report (in long date format, short date, and so on.)
32-bit DLL	U2LTDATE.DLL
16-bit DLL	UFLTDATE.DLL
Adds functions	NumberToDate(Number)

Formula Syntax

`NumberToDate(x)`	Where "x" is a NUMERIC value in the format YYYYMMDD, this function will convert it into Crystal Reports Date format: Date(YYYY,MM,DD).

Examples

```
NumberToDate(20001231)
```

This returns Date(2000,12,31)

While you can sometimes use numbers in the format YYMMDD, they cannot be formatted in CR with a long year.

For example,

```
NumberToDate(991030)
```

returns Date(1999,10,30). Often, the default displays format on the report for dates is MM/DD/YY, so this formula returns 10/30/99.

However, if you format this formula field to display in MMM DD, YYYY format, this formula returns October 30, 99.

Years 00 to 09 provide an even greater formatting challenge: For example, NumberToDate(020101) will displays in MM/DD/YY format as 01/01/2, and displays in MMM DD, YYYY format as January 1, 2

The solution? Use a 4-digit number for year with the NumberToDate function.

String functions

This sections describes additional functions to:

- Capitalize the first letter of every word in a string (Uflcaps.exe).
- Search for a particular word or phrase within a string, and replace it with another one (Uflrepl.exe).
- Extract a portion or ‘piece’ of a string, when the ‘piece’ is separated by a particular character (Uflstr.exe).
- Determine the starting position of a string within a larger string (Uflstring.exe)
- Extract a user-specified string from within another string, return the first characters of each word within a string, and returns a user-specified number of words from a string (Ufltech1.exe)

Title Caps UFL

Filename	UFLCAPS.EXE Compatible with versions: 4.0 and higher
Description	Capitalizes the first character of a string after a space, tab or hyphen.
32-bit DLL	U2LCAPS.DLL
16-bit DLL	UFLCAPS.DLL
Adds functions	LeadingCaps(string)

Formula Syntax

```
LeadingCaps(x)
```

Where "x" is a string that you wish to have each "word" converted to title case (the first letter of each word is put in uppercase). This function considers everything separated by spaces, tabs or hyphens to be "words".

Examples

```
LeadingCaps("tHiS IS THE uflcaps-conVERTed strinG.")
```

This returns the string: "This Is The Uflcaps-Converted String."

Notice that letters following a space, tab or hyphen character are capitalized.

Search and Replace UFL

Filename	UFLREPL.EXE Compatible with versions: 4.5 and higher
Description	This function searches the string "X" for the string "Y" and replaces every instance of "Y" with "Z". If you want "Y" to be searched with case-sensitivity, ensure the "Boolean" says True (no quotes).
32-bit DLL	U2LREPL.DLL
16-bit DLL	UFLREPL.DLL
Adds functions	SearchAndReplace (X,Y,Z,Boolean)

Formula Syntax

This function searches the string "X" for the string "Y" and replaces every instance of "Y" with "Z". If you want "Y" to be searched with case-sensitivity, ensure the "Boolean" says True (no quotes).

Examples

```
SearchAndReplace ("This is my dog.", "DOG", "CAT", False)
```

Returns the string "This is my CAT."

```
SearchAndReplace ("I am blue, BLUE, Blue", "blue", "red",
True)
```

This formula returns the string "I am red, BLUE, Blue"

NOTE	The "searchandreplace" function in version 8 and 8.5 does not need to be used this is because the "replace" function provides this functionality.

String Token UFL

Filename	UFLSTR.EXE Compatible with versions: 4.0 and higher
Description	This adds a function allowing you to return a "token" or "piece" of a string, where each piece/token is separated by a user-specified character.
32-bit DLL	U2LSTR.DLL
16-bit DLL	UFLSTR.DLL
Adds functions	StrTok (string to be parsed, separator, token number)

Formula Syntax

```
StrTok (x, y, z)
```

Where "x" is the string you are parsing, "y" is your separator that is between each token and "z" is the numeric representing which "token" you want to return.

Examples

```
{string1} = "first.second.third.fourth"
```

```
StrTok ({string1}, ".", 0)
```

Returns Error message stating system is begins with '1'

```
StrTok ({string1}, ".", 1)
```

Returns "first"

```
StrTok ({string1}, ".", 2)
```

Returns "second"

```
StrTok ({string1}, ".", 3)
```

Returns "third"

```
StrTok ({string1}, ".", 4)
```

Returns "fourth"

```
StrTok ({string1}, ".", 5)
```

Returns Error message stating only X tokens available.

NOTE	This function requires that you KNOW how many "tokens" are in the string you are parsing. Otherwise, your formula will return the errors listed above.

Search String UFL

Filename	UFLSTRNG.EXE Compatible with versions: 4.x or higher
Description	Additional functions that return the starting position of a string within another string or string of characters before or after a certain character.
32-bit DLL	U2LSTRNG.DLL
16-bit DLL	UFLSTRNG.DLL
Adds functions	SearchString (StringField, Character) StringBeforeChar (StringField, Character) StringAfterChar (StringField, Character)

Formula Syntax

Where "x" is the string you want to search, and "y" is the character you are looking for.

`SearchString (x,y)`	This function will return the position of the first occurrence of "y" in "x" as a NUMBER, or if no match found, will return the LENGTH of "x".
`StringBeforeChar (x,y)`	This function will return the STRING of characters found before the first occurrence of "y" in "x". If no match is found, the entire string "x" will be returned.

`StringAfterChar (x,y)`	This function will return the STRING of characters found after the first occurrence of "y" in "x". If no match is found, the entire string "x" will be returned.

Examples

Where StringField = "Tummonds*Fred" and Character = "*"
SearchString("Tummonds*Fred","*")

Returns a NUMERIC of 9.0

`StringBeforeChar("Tummonds*Fred","*")`

Returns the STRING "Tummonds"

`StringAfterChar("Tummonds*Fred","*")`

Returns the STRING "Fred"

String and Square Root UFLs

Filename UFLTECH1.EXE
Compatible with versions: 4.0 and higher

Description This file contains several 32-bit and 16-bit functions:

- Square Root calculates the square root of a numeric value
- StripString extracts a user-specified string from within another string;
- GetInitials returns the first characters of each word within a string
- GetWord returns a user-specified number of words from a string.

32-bit DLL U2LTECH1.DLL

16-bit DLL UFLTECH1.DLL

Adds functions SquareRoot (numberfield)
StripString (string, string)
GetInitials (string)
GetWord (string, number)

Formula Syntax

`SquareRoot (x)`	Where "x" is a positive number, the function returns the square root of "x".
`StripString (x, y)`	Where "x" is a string and "y" is the string you want remove all occurrences of from "x".
`GetInitials (x)`	Where "x" is a string containing "words" separated spaces, this function will return all the FIRST of each word in "x".
`GetWord (x, y)`	Where "x" is a string containing "words" separated by spaces or commas, and "y" is the NUMBER of the word" you want to return.

Examples

`SquareRoot (16)`

Returns 4.00

`SquareRoot (73)`

Returns 8.54

`SquareRoot (-8)`

Returns Error

`StripString ("This is a test.", "is")`

Returns "Th a test."

`GetInitials ("John J. Doe")`

Returns "JJD"

`GetInitials ("University of British Columbia")`

Returns "UoBC"

`GetWord ("John J. Doe", 3)`

Returns "Doe"

`GetWord ("Doe,J John", 2)`

Returns "J"

Mathematical functions

This section describes additional functions to:

- calculate exponential values and display a number in scientific notation
- convert hexadecimal, binary, log and fractions to numbers and vice versa
- calculate trigonometric values: sine, cosign, tangent and arctangent
- calculate the square root of a number
- calculate running totals, counts, and other summaries on record-by-record basis.

Exponent UFL

Filename	UFLEXPO.EXE Compatible with versions: 4.0 and higher
Description	Adds new functions allowing you to return exponential numbers and numbers in scientific format.
32-bit DLL	U2LEXPO.DLL
16-bit DLL	UFLEXPO.DLL

Adds functions	EXPO(x,y)
	ScNotn(x,y)

Formula Syntax

`EXPO(x,y)`	Will return a number of "x" to the power of "y"
`ScNotn(x,y)`	Will return a string of "x" in scientific notation with "y"-1 significant digits.

Examples

`EXPO(2,4)`	Returns 16.0
`ScNotn(2,4)`	Returns 2.000E+00

Math UFL

Filename	UflMath.exe Compatible with versions: 7.0 32-bit
Description	User Function Library (UFL) containing several math conversion functions including Hexadecimal digits, fractions, logarithms, and Trigonometric functions. These functions can be used to extend the Crystal Reports formula language.
32-bit DLL	CRUFLMath.dll
16-bit DLL	**Only available in 32-bit**
Adds functions	NumberToHex (Number) HexToNumber (Hex) BinaryToNumber (Binary) NumberToFraction (Number) NumberToLog (Number) Sine (Number) Cosine (Number) Tangent (Number) Arctangent (Number)

Installation

To include these functions into the Additional Functions Section of the Crystal Formula Editor:

1. Copy CRUFLMath.dll to the <system drive>\<system root>\Crystal folder

   (i.e. c:\Windows\Crystal)
2. Register this DLL into the Microsoft Windows registry (the UFL must be registered before it's functions are made available).

To register the DLL, ensure the registry utility Regsvr32.exe exists on your machine (this normally resides in the <system> folder). From the Windows Desktop click the Start button and select Run.

Type the following:

```
Regsvr32 <system drive>\<system root>\Crystal\crUFLMath.dll
```

(i.e. Regsvr32 c:\Windows\Crystal\crUFLMath.dll)

Formula Syntax

NumberToHex (Number)	returns a Hex String.
HexToNumber (“Hex”)	returns a number. Hex must be a string.
BinaryToNumber (“Binary”)	returns a number. Binary must be a string.
NumberToFraction (Number)	returns a fraction as a String.
NumberToLog (Number)	returns a number as the Logarithm of a number.
Sine (Number)	returns the sine of a number.
Cosine (Number)	returns the cosine of a number.
Tangent (Number)	returns the tangent of a number.
Arctangent (Number)	returns the arctangent of a number.

Square Root UFL

Filename	UFLSQRT.EXE Compatible with versions: 4.0 and higher
Description	This adds the function to allow you to show square roots of positive numeric fields on your reports.
32-bit DLL	U2LSQRT.DLL
16-bit DLL	UFLSQRT.DLL
Adds functions	SquareRoot(x)

Formula Syntax

`SquareRoot(x)` Where "x" is any POSITIVE number.

Examples

`SquareRoot(16)` Returns 4.00

Running Total Function UFL

Filename	UF5RT.EXE ** For use with Crystal Reports version 5 and higher. **
Description	Additional functions to calculate running total variables without using the long formula method.
32-bit DLL	U25RT.DLL
16-bit DLL	UF5RT.DLL

Adds functions	RTReset("VarName")
	RTSet("VarName", NumberField, "WhoString")
	RTGet("VarName", "FunctionName")
	RTGetWho("VarName", "FunctionName")

Other

This sections describes additional functions to:

- convert ASCII characters to reveal their number codes and vice-versa.
- convert numbers to bar codes
- convert OEM strings to ANSI and vice-versa
- return a string containing the value of the currently selected DOS environment variable.
- convert metric measurements to imperial and vice-versa
- create your own UFL
- pass data from a Subreport to a main report, and vice-versa..

ASCII Conversion UFL

Filename	UFLASC.EXE Compatible with versions: 4.0 and higher	
Description	Additional functions to convert characters to ASCII number codes and ASCII number codes into characters.	
32-bit DLL	U2LASC.DLL	
16-bit DLL	UFLASC.DLL	
Adds functions	ASC(x)	Returns the ASCII code of the character specified as "x"
	CHR(x)	Returns the ASCII character of the number specified as "x"

Formula Syntax

`ASC(x)`	Where "x" is any single character of the ASCII character set, will return a NUMBER representing the ASCII code of that character.
`CHR(x)`	Where "x" is a NUMBER from 1 to 255, will return the Windows ASCII CHARACTER represented by that code number.

Examples

`ASC("A")`	Returns the number 65.0
`CHR(199`	Returns the character Ç

The CHR(x) function is especially useful in formulas where you are wanting to insert a carriage return (enter) between two or more strings. For example, this formula:

```
@MultiLines
"This is line one." + CHR(13) + "This is line two."
```

returns two lines:

This is line one.

This is line two.

13 is the ASCII code for 'carriage return', so inserting a CHR(13) between two strings ensures each string prints on separate line.

NOTE	The above functions work for ASCII codes 1 - 255 based on the Window ASCII character set which is different than the DOS ASCII character set.

OEM / ANSII Conversion UFL

Filename	UFLCONV.EXE Compatible with versions: 4.0 and higher
Description	Additional functions to convert ANSI strings to OEM and OEM to ANSI
32-bit DLL	**16bit ONLY**
16-bit DLL	UFLCONV.DLL
Adds functions	StrANSI2OEM (x) StrOEM2ANSI (x)

Formula Syntax

`StrANSI2OEM (x)` This will convert an ANSI string "x" to an OEM string

`StrOEM2ANSI (x)` This will convert an OEM string "x" to an ANSI string

DOS Variable UFL

Filename	UFLENV.EXE Compatible with versions: 4.0 and higher
Description	Adds a new function for 16bit only that allows you to return a string containing the value of the currently selected environment variable.
32-bit DLL	**16 bit only**
16-bit DLL	UFLENV.DLL
Adds functions	GetEnvString(Environment String)

Formula Syntax

`GetEnvString(x)`	Where "x" is the DOS environment variable in UPPERCASE that you want to return. (Such as "PATH","SET", and so forth.)

Examples

`GetEnvString("PATH")`	Returns the string containing your PATH statement: "PATH=C:\WINDOWS;C:\;C:\DOS;C:\CRW;"

Metric and Imperial Conversion UFL

Filename	UFLMetricConv.exe Compatible with versions: 7.0 32-bit
Description	User Function Library (UFL) containing several Metric to Imperial and Imperial to Metric conversion functions. These functions can be used to extend the Crystal Reports formula language.
32-bit DLL	crUFLConv.dll
16-bit DLL	**Only available in 32-bit**
Adds functions	CentimetersToInches (Centimeters) InchesToCentimeters (Inches) KilometersToMiles (Kilometers) MilesToKilometers (Miles) CelciusToFahrenheit (Celsius) FahrenheitToCelcius (Fahrenheit) GramsToOunces (Grams) OuncesToGrams (Ounces) KilogramsToPounds (Kilograms) PoundsToKilograms (Pounds)

Installation:

To include these functions into the Additional Functions Section of the Crystal Formula Editor:

1. Copy CRUFLConv.dll to the <system drive>\<system root>\Crystal folder

   (i.e. c:\Windows\Crystal)
2. Register this DLL into the Microsoft Windows registry (the UFL must be registered before it's functions are made available).

To register the DLL, ensure the registry utility Regsvr32.exe exists on your machine (this normally resides in the <system> folder). From the Windows Desktop click the Start button and select Run. Type the following:

```
Regsvr32 <system drive>\<system root>\Crystal\crUFLConv.dll
```

(i.e. Regsvr32 c:\Windows\Crystal\crUFLConv.dll)

Formula Syntax

`CentimetersToInches (Centimeters)` converts a number representing centimeters to a number representing inches.

`InchesToCentimeters (Inches)` converts a number representing inches to a number representing centimeters.

`KilometersToMiles (Kilometers)` converts a number representing Kilometers to a number representing Miles.

`MilesToKilometers (Miles)` converts a number representing Miles to a number representing Kilometers.

`CelsiusToFahrenheit (Celsius)` converts a number representing Celsius to a number representing Fahrenheit.

`FahrenheitToCelsius (Fahrenheit)` converts a number representing Fahrenheit to a number representing Celsius.

`GramsToOunces (Grams)` converts a number representing Grams to a number representing Ounces.

`OuncesToGrams (Ounces)` converts a number representing Ounces to a number representing Grams.

`KilogramsToPounds (Kilograms)` converts a number representing Kilograms to a number representing Pounds.

`PoundsToKilograms (Pounds)` converts a number representing Pounds to a number representing Kilograms.

UFL Template

Filename	UFLSKEL.EXE Compatible with versions: 4.0 and higher of the Professional version only
Description	This is a sample DLL and code to show the basic method of creating your own UFL function DLL's with C++. Includes sample C++ code to create 16 and 32-bit DLLs.
32-bit DLL	U2LSKEL.DLL
16-bit DLL	UFLSKEL.DLL
Adds functions	Design Your Own!

Formula Syntax

You Decide!

Examples

You Decide!

For more information on creating your own UFLs, search under ‘Creating User Defined Functions…” in the index of Crystal Reports online Developer’s Help.

Store and Fetch UFL

Filename	UF5STORE.EXE Compatible with versions: 5 and 6.
Description	These DLLs add a number of new functions that allow you to pass string, numeric, date, currency and Boolean variables back and forth between the Main Report and any number of Subreports.
32-bit DLL	U25STORE.DLL
16-bit DLL	UF5STORE.DLL

Adds functions	StoreNumberVar(X, Y)
	StoreStringVar(X, Y)
	StoreDateVar(X, Y)
	StoreCurrencyVar(X, Y)
	StoreBooleanVar(X, Y)
	FetchNumberVar(X)
	FetchStringVar(X)
	FetchDateVar(X)
	FetchCurrencyVar(X)
	FetchBooleanVar(X)

Since the introduction of subreports in Crystal Reports 5.0, users have wanted a way to pass information between the subreport and the main report. UFLSTORE addresses this issue by creating a global memory variable; a variable whose contents are available to both the main report and any subreport contained within it.

Practical uses for UFLSTORE are numerous and most of them relate to subreports. If you have created a subreport and would like to use the information that it generates in the main report, then UFLSTORE is the way to do it. The exciting thing is that the information stored is available not only to the main report but to all subreports contained with in it. A few ideas from Tech Support:

- Create a Grand Total that adds all totals from the subreport and main report. This is a good way for combining totals that come from non-related databases.
- Create an index for your main report. With a good working knowledge of your database you could create an index that will tell the user which page certain information (groups, totals etc) is printed on.

Formula Syntax

`@Store Formula`

The first step is to place the information into the memory variable by using the store function that matches the type of information being stored.

i.e. StoreNumberVar(x,y) for numeric fields,

StoreStringVar(x,"y") for string fields - note x must be a string.

`@Fetch Formula`

The second step is to retrieve the information using the corresponding fetch function and variable name

`@Store Example`

StoreNumberVar("SubreportTotal",1000)

This formula stores the variable called 'SubreportTotal' with a value of 1000 into memory.

`@Fetch Example`

`FetchNumberVar("SubreportTotal")`

This formula fetches the variable created in the @Store formula, and returns the value that was assigned to the variable in the @Store formula.

To ensure the @Fetch formula successfully retrieves the variable declared & assigned a value in the @Store formula, you must place @Fetch in a section beneath the section containing @Store.

To store a value in a subreport **and retrieve it in the main report**:

- In the subreport, create and insert @Store formula to store the value.
- In the main report, create and insert @Fetch in a section beneath the section containing the subreport .

To store a value in the main report **and** retrieve it in the subreport:

- In the main report, create and insert @Store.
- In the subreport, create and insert @Fetch.
- In the main report, place the subreport in a section beneath the section containing @Store.

NOTE	For more information about using Store and Fetch functions, please go to http://support.crystaldecisions.com/docs and search for the file Uflstore.pdf (the Store and Fetch Functions technical brief).

Examples

`StoreNumberVar(X, Y)`

Usage: Stores a numeric value Y to X. If a formula field contains this function, it will return the numeric value Y.

Example: StoreNumberVar("gTotal", 1000).

Returns: 1000

`StoreStringVar(X, Y)`

Usage: Stores a string value Y to key X. If a formula field just contains this function, it will return a string value Y.

Example: StoreStringVar("My Name", "Beanie")

Returns: Beanie

`StoreDateVar(X, Y)`

Usage: Stores a date value Y to key X. If a formula field just contains this function, it will return a date value Y.

Example: StoreDateVar("MyBDay", Date(1970,04,02))

Returns: April 2, 1970

`StoreCurrencyVar(X, Y)`

Usage: Stores a string value Y to key X. If a formula field just contains this function, it will return a string value Y. This is same as using SaveNumberVar().

Example: StoreCurrencyVar("gTotal", 1000)

Returns: $1,000.00

`StoreBooleanVar(X, Y)`

Usage: Stores a True/False value Y to key X. If a formula field just contains this function, it will return a string value Y.

Example: StoreBooleanVar("Gender Is Male", True)

Returns: True

`FetchNumberVar(X)`

Usage: Given the key X, it will return a numeric value that is stored under this key.

Example: FetchNumberVar("gTotal")

Returns: 1000

`FetchStringVar(X)`

Usage: Given the key X, it will return a string value that is stored under this key.

Example: FetchStringVar("My Name")

Returns: Beanie

`FetchDateVar(X)`

Usage: Given the key X, it will return a date value that is stored under this key.

Example: FetchDateVar("MyBDate")

Returns: April 2, 1970

`FetchCurrencyVar(X)`

Usage: Given the key X, it will return a currency value that is stored under this key.

Example: FetchCurrencyVar("gTotal")

Returns: $1,000.00

`FetchBooleanVar(X)`

Usage: Given the key X, it will return a Boolean value that is stored under this key.

Example: FetchBooleanVar("Gender is Male")

Returns: True

Tips when using the UFLSTORE functions:

- Use WhilePrintingRecords in all formulas that use the store and fetch functions. This will make Crystal Reports evaluate the formulas during the same evaluation time.
- The formula that fetches the variable MUST NOT be in the same section that contains the subreport. It must be in a section BELOW the subreport section. This is to make sure that the subreport where a value is saved into the variable gets evaluated first than the formula that fetches the variable.
- If the subreport is suppressed or if the query of the subreport does not return any records, the formulas that store into the variable will not be evaluated. This means the variable never gets created, and this will cause a "Variable not found" error if you try to fetch the variable. To avoid this, create a formula in the main report that initializes the variable to zero, and insert this into a section above the section that contains the subreport. This will cause the variable to be created even if the formula in the subreport is not evaluated.

Contacting Crystal Decisions for Technical Support

We recommend that you refer to the product documentation and that you visit our Technical Support web site for more resources.

Self-serve Support:

http://support.crystaldecisions.com/

Email Support:

http://support.crystaldecisions.com/support/answers.asp

Telephone Support:

http://www.crystaldecisions.com/contact/support.asp

Appendix: Alphabetical List of Additional Functions

If you know the name of the particular additional function you need, but don't know the name of the EXE file containing the EXE file that you need to download, use this index. Once you know the EXE filename, you can then go to http://support.crystaldecisions.com/downloads and search for the file.

Function	Filename
A	
Arctangent (Number)	UFLMATH.EXE (32-bit only)
Asc ("Character")	UFLASC.EXE
B	
BinaryToNumber ("Binary string")	UFLMATH.EXE (32-bit only)
Btime (Decimal Number)	UFLBTIME.EXE
C	
CelciusToFahrenheit (Celsius)	UFLMETRICCONV.EXE (32-bit only)
CentimetersToInches (Centimeters)	UFLMETRICCONV.EXE (32-bit only)
Chr (Number)	UFLASC.EXE
Cosine (Number)	UFLMATH.EXE (32-bit only)
CRTimeFromSecsSinceMidnight (Seconds)	UFLSSM.EXE
D	
DateAdd (Interval, Number, Date)	UFLDATEADD.EXE (32-bit only)
DateTimeDiff (DateTime, DateTime)	UFLDTDIF.EXE
DateTimeTo2000 (DateTime, Number)	UF52000.EXE
DateTo2000 (Date, Number)	UF52000.EXE
DateToJulian (Date)	UFLJUL.EXE
DTSTo2000 ("DateTimeString", Number)	UF52000.EXE
DTSTo2000("DateTimeString",Number)	UFL2000.EXE
DayOfYear(Date)	UFLDATE.EXE
E	
Expo (Number, Number)	UFLEXPO.EXE
F	
FahrenheitToCelsius (Fahrenheit)	UFLMETRICCONV.EXE (32-bit only)
FetchBooleanVar ("VarName")	UF5STORE.ZIP
FetchCurrencyVar ("VarName")	UF5STORE.ZIP
FetchDateVar ("VarName")	UF5STORE.ZIP
FetchNumberVar ("VarName")	UF5STORE.ZIP
FetchStringVar ("VarName")	UF5STORE.ZIP
G	
GetEnvString ("DOS Environment Variable")	UFLENV.EXE (16 bit only)
GetInitials ("String")	UFLTECH1.EXE
GetWord ("String", Number)	UFLTECH1.EXE
GramsToOunces (Grams)	UFLMETRICCONV.EXE (32-bit only)

H

HexToNumber("Hex string")	UFLMATH.EXE (32-bit only)
HHMMFromSecsSinceMidnight (Seconds)	UFLSSM.EXE
HHMMPMFromSecsSinceMidnight (Seconds)	UFLSSM.EXE
HHMMSSFromSecsSinceMidnight (Seconds)	UFLSSM.EXE
HHMMSSPMFromSecsSinceMidnight (Seconds)	UFLSSM.EXE
HoursFromSecsSinceMidnight (Seconds)	UFLSSM.EXE

I

InchesToCentimeters (Inches)	UFLMETRICCONV.EXE (32-bit only)

J

JulianToDate (Julian Date)	UFLJUL.EXE

K

KilogramsToPounds (Kilograms)	UFLMETRICCONV.EXE (32-bit only)
KilometersToMiles (Kilometers)	UFLMETRICCONV.EXE (32-bit only)

L

LeadingCaps ("String")	UFLCAPS.EXE

M

MilesToKilometers (Miles)	UFLMETRICCONV.EXE (32-bit only)
MinutesFromSecsSinceMidnight (Seconds)	UFLSSM.EXE

N

NumberToDate (Number)	UFLTDATE.EXE
NumberToHex (Number)	UFLMATH.EXE (32-bit only)
NumberToFraction (Number)	UFLMATH.EXE (32-bit only)
NumberToLog (Number)	UFLMATH.EXE (32-bit only)

O

OuncesToGrams (Ounces)	UFLMETRICCONV.EXE (32-bit only)

P

PDXTimeToCRTime (PDXTimeField)	UFLPDXTM.EXE
PoundsToKilograms (Pounds)	UFLMETRICCONV.EXE (32-bit only)

R

RTGet ("VarName", "FunctionName")	UF5RT.EXE
RTGetWho ("VarName", "FunctionName")	UF5RT.EXE
RTReset ("VarName")	UF5RT.EXE

RTSet ("Varname", NumberField, "WhoString")	UF5RT.EXE
S	
ScNotn (Number, Number)	UFLEXPO.EXE
SearchAndReplace (“String x”, “String y”, “String z”, Boolean)	UFLREPL.EXE
SearchString (“String”, “Character”)	UFLSTRNG.EXE
SecondsFromSecsSinceMidnight (Seconds)	UFLSSM.EXE
Sine (Number)	UFLMATH.EXE (32-bit only)
Sqrt (Positive Number)	UFLSQRT.EXE
SquareRoot (Positive Number)	UFLTECH1.EXE
StoreBooleanVar (“VarName”, Boolean)	UF5STORE.EXE
StoreCurrencyVar (“VarName”, Currency)	UF5STORE.EXE
StoreDateVar (“VarName”, Date)	UF5STORE.EXE
StoreNumberVar (“VarName”, Number)	UF5STORE.EXE
StoreStringVar (“VarName”, “String”)	UF5STORE.EXE
StrANSI2OEM (“ANSI String”)	UFLCONV.EXE (16 bit only)
StringAfterChar (“String”, “Character”)	UFLSTRNG.EXE
StringBeforeChar (“String”, “Character”)	UFLSTRNG.EXE
StripString” (“String”, “String”)	UFLTECH1.EXE
StrOEM2ANSI (“OEM String”)	UFLCONV.EXE (16 bit only)
StrTok (“String to be parsed”, “Separator”, Token Number)	UFLSTR.EXE
T	
Tangent (Number)	UFLMATH.EXE (32-bit only)
W	
WeekOfYear(Date, Boolean)	UFLSSM.EXE