crystal decisions™

Crystal Reports 5.x, 6.x, 7.x and 8.x

Date and DateTime functions and conversions

Overview

This document outlines the formula functions available for use with DateTime fields and DateTime strings in Crystal Reports versions 5, 6, 7, and 8.x. There are sample formulas that demonstrate these functions in extracting time and date information and converting date fields stored as a number or string to proper date or DateTime values.

Contents

Introduction

This document provides detailed information and sample formulas for use with DateTime fields and strings in Crystal Reports version 5, 6, 7, 8, and 8.5. This paper is broken down into a various sections. You will find information on:

- Explaining how DateTime fields are read in Crystal Reports
- Using DateTime functions specific to Crystal Reports 8 and higher, which includes information on:
  - DateTime Conversions
  - Time Conversions
- Using DateTime functions available in Crystal Reports version 5, 6, 7, 8 and 8.5, which includes information on:
  - DateTime string Conversion
  - DateTime field Conversion
- Using available additional DateTime functions in Crystal Reports version 5, 6, 7, 8 and 8.5
- Creating formulas that convert fields and strings to a Date type
- Creating a formula that calculates the difference between two DateTime fields

DateTime fields read in Crystal Reports

DateTime fields are stored in databases various ways. To find out how Crystal Reports version 5, 6, 7, 8, and 8.5 reads DateTime fields, you can browse the field data. Generally, they come in the following formats:

Assume a Date and Time of January 31, 1998 at 12:30:30PM

Data Type	Format	Appearance when

		browsing field in CR
Date	Date (yyyy,mm,dd)	Date (1998,01,31)
DateTime	Date (yyyy,mm,dd,hh,mm,ss)	Date (1998,01,31,12,30,30)
DateTime String	yyyy/mm/dd hh:mm:ss.ss	1998/01/31 12:30:30.00
Date String	mmddyy or any format	01/31/98
Numeric	yyyymmdd or any format	19980131
Numeric	nnnnn	35824 (Julian Date)

Converting DateTime fields in Crystal Reports

To convert a DateTime field in Crystal Reports, complete the following steps:

1. From the 'File' menu, select 'Report Options'.
2. Select the option applicable to the version of Crystal Reports you are working with.

- In Crystal Reports 5, select 'Convert DateTime to Date'. If you do not want to convert a DateTime field to Date format, clear this option if it is selected.
- In Crystal Reports version 6, 7, 8 and 8.5, select on of the following options:
  - 'Convert DateTime to Date'
  - 'Convert DateTime to DateTime'
  - 'Convert DateTime to String'.

If you do not want to convert to any of these formats, clear the options.

Toggling these options changes the way Crystal Reports reads the DateTime fields.

NOTE:	In Crystal Reports 5.x, none of the Crystal Reports database drivers read a DateTime field as DateTime. In Crystal Reports 6.x, only ODBC database drivers and some databases such as Informix and Sybase can read DateTime fields as DateTime fields in Crystal Reports. This ONLY works if the option "Convert Date/Time to Date/Time" is selected. In Crystal Reports 7 and higher, as long as the database has DateTime field and the "Convert DateTime to DateTime" option is selected, all DateTime fields should be read as true DateTime values.

DateTime and Time functions in Crystal Reports 8.x

Crystal Reports 8 and higher have a few new functions to convert dates, date-times, and time. These new functions were added to allow converting many different formats to dates, date-times and time. These functions are:

- Cdate()
- Cdatetime()
- Ctime()

Each of these functions work as a "one function fits all" as opposed to the more specific functions available in Crystal Reports 7 and lower.

NOTE:	The more specific functions available in Crystal Reports 7 and lower can still be used in Crystal Reports 8 and higher. It is easier to use the Cdate(), Cdatetime(), and Ctime() functions in Crystal Reports 8 and higher.

Date and DateTime conversion functions

In Crystal Reports 8 and higher, some new functions were added that convert many different formats to dates, date-times or time. In the following section you will find information on the Cdate() functions available in Crystal reports 8 and higher.

CDate()

The Cdate() function has the ability to convert four data types to true date format. The functions are:

- CDate(number)
- CDate(string)
- CDate(dateTime)
- CDate(YYYY,MM,DD)

CDate(number)

In the CDate(number) function, the "number" argument represents the number of days since December 30, 1899.

For example:

```
CDate(50)
```

returns February 18, 1900.

CDate(string)

In the CDate(string) function, the "string" argument converts strings of varying formats to dates.

For example:

```
CDate("August 28, 2001")
CDate("2001/08/28")
CDate("2001-08-28")
CDate("08/28/01")
```

all return August 28, 2001.

CDate(datetime)

In the Cdate(datetime) function, the "datetime' argument converts a DateTime field to a date by ignoring the time portion of the DateTime field.

CDate(YYYY, MM,DD)

In the Cdate(YYYY,MM,DD) function, the "YYYY,MM,DD" argument converts a four digit year, month number and day number into a date format.

NOTE:	If Cdate encounters a format it does not understand, such as a null value, the formula will fail. In order to error trap for this, use the IsDate() function. IsDate({field}) returns a true or false based on whether the {field} is a recognized format that Cdate can convert. If IsDate({field}) then Cdate({field}) else Date(0,0,0) IsDateTime() and IsTime() can be used similarly with CdateTime and Ctime respectively.

CDateTime()

The CDateTime() function has the ability to convert five data types to true date format. The functions are:

- CDateTime(number)
- CDateTime(string)
- CDateTime(Date)
- CDateTime(Date,Time)
- CDateTime(YYYY,MM,DD,hours,minutes,seconds)

CDateTime(number)

In the CDateTime(number) function, the "number' argument represents the number of days since December 30, 1899.

For example:

```
CDateTime(50.25)
```

returns February 18, 1900 6:00 am.

CDateTime(string)

In the CDateTime(string) function, the "string" argument converts strings of varying formats to date-times.

For example:

```
CDateTime("August 28, 2001")
```

returns August 28, 2001 12:00 am. Because, a time was not specified, 12:00 am is returned.

```
CDateTime("2001/08/28 6:00:38")
```

returns August 28, 2001 6:00:38 am.

The IsDateTime() function can be used to check whether the format of the string is valid for the CdateTime function.

CDateTime(date)

In the CDateTime(date) function, the 'date' argument converts a date field to a DateTime by appending a 12:00:00 am timestamp to the date.

For example:

```
CDateTime("2001/08/28")
```

returns 08/28/2001 12:00:00AM

CDateTime(date,time)

In the CDateTime(Date,Time) function, the "Date,Time" argument concatenates a date and a time to give a DateTime field.

CDateTime(YYYY, MM,DD, hours, minutes, seconds)

In the CDateTime(YYYY, MM,DD, hours, minutes, seconds) function, the "YYYY, MM,DD, hours, minutes, seconds" argument creates a DateTime field from 6 numeric arguments representing four digit year, month, day, hours, minutes, and seconds. If the hours, minutes, and seconds are not included, a 12:00 am time is the default.

For example

```
CDateTime(2001,10,8,8,11,32)
```

returns 10/08/2001 8:11:32AM

Time Conversion Functions

In Crystal Reports 8 and higher, some new functions were added that convert many different formats to dates, date-times or time. In the following section you will find information on the CTime functions available in Crystal reports 8 and higher.

CTime()

- CTime(number)
- CTime(string)
- CTime(dateTime)
- CTime(hours,minutes,seconds)

CTime(number)

In the CTime(number) function, the "number" argument returns a time value based on the fractional portion of a number.

For example

```
CTime(0.5)
```

returns 12:00 pm

```
CTime(0.75)
```

returns 6:00 pm

CTime(1.75) also returns 6:00 pm because only the fractional portion affects the time.

CTime(string)

In the CTime(string) function, the 'string' argument converts strings of varying formats to times.

For example:

```
CTime("August 28, 2001 4:16:04 pm")
CTime("16:16:04")
```

Will both return 4:16:04 pm.

CTime(datetime)

In the CTime(datetime) function, the "datetime' argument returns the time portion of a datetime field.

CTime(hours,minutes,days)

In the CTime(hours,minutes,days) function, the "hours,minutes,days" argument works like the Time() function to convert numbers representing hours, minutes, and seconds to a time format.

For example

```
CTime(15,33,59)
```

returns 3:33:59 PM.

Conversion Functions for various versions of Crystal Reports

DateAdd Functions for Crystal Reports 6, 7 & 8.x

DateAdd(intervalType, nIntervals,{DateTimeField})

The DateAdd function has the ability to add or subtract a number of specified time intervals and outputs a valid DateTime field. The function looks like the following:

```
DateAdd(intervalType, nIntervals,{DateTimeField})
```

The DateAdd function is installed automatically in Crystal Reports 8 and higher.

NOTE	To obtain the DateAdd function for Crystal reports 6 and 7, you can download the additional function called Ufldateadd.exe, from the Crystal Decisions support site at: http://support.crystaldecisions.com/downloads

For Example:

```
DateAdd("s",7200,#October 1, 2001#)
```

The "s" argument in this function adds 7200 seconds (2 hours) to Oct 1,2001 12:00 am to output 10/01/2001 2:00:00AM.

```
DateAdd("m",-1,#March  31, 2001#)
```

The "m" argument in this function subtracts one month from March 31, 2001 returning a result of February 28, 2001 12:00:00 AM.

```
DateAdd("q",3,Date(2001,2,28))
```

The "q" argument in this function adds 3 quarters (or nine months) to February 28, 2001 returning a result of 11/28/2001 12:00:00AM

The following table lists valid Interval Types for the first argument of the DateAdd function.

IntervalType value	Description
yyyy	Year
Q	Quarter (3-month period)
M	Month
Y	Day of year
D	Day
W	Weekday
Ww	Week (7-day period)
H	Hour
N	Minute
S	Second

DateSerial Function for Crystal Reports 8.x

DateSerial(yyyy,mm,dd)

The DateSerial function is similar to the CDate function except that it allows the day numbers to be > 31 and month numbers > 12. This is very useful when creating a record selection formula.

For example:

```
DateSerial(2001,13,1)
```

returns a date of Jan 1 2002.

```
DateSerial(year(currentdate),
month(currentdate),day(currentdate) - 5)
```

The "-5" argument returns a date 5 days earlier than the current date. If you tried using the same formula with Cdate(), you would get an error when the day portion was less than 6.

If you wanted to return the last day of the last month, you could use the following formula:

```
DateSerial(year(currentdate),month(currentdate),1 - 1)
```

This formula takes the first day of the current month and subtracts one day. This will take into account whether the previous month has 28,29,30, or 31 days.

Crystal Reports 5, 6 & 7 DateTime Conversion

The following functions are also available in Crystal Reports version 8.x but for the most part can be replaced with one of the Cdate functions.

DateTime Strings

DTSToDate({DateTime String})

The DTSToDate({DateTime String}) function extracts the date from a DateTime string field.

NOTE:	The DateTime string must be in the format "'YYYY/MM/DD HH:MM.SS". The results can then be used in comparisons for selection and computation formulas.

For example:

```
DTSToDate ({table.datetimestring})+1
```

This adds 1 day to the resulting date.

The following is a sample record selection formula:

```
DTSToDate ({table.datetimestring}) <
Date(1998,01,31)
```

This prints records earlier than January 31, 1998.

DTSToTimeString (DateTime String)

The DTSToTimeString (DateTime String) function takes a DateTime string and extracts the time portion of the string. Because it returns a string, it can be treated the same as any other string field, but the result cannot be used to do comparison or addition formulas.

For example:

The following is an example used in a record selection formula :

```
DTSToTimeString ({table.datetimestring}) =
DTSToTimeString ("1994/12/12 05:21:35.00")
```

This returns records with the matching time string of "05:21:35.00"

DTSToSeconds(DateTime String)

The DTSToSeconds(Datetime String) function takes the DateTime string and extracts the number of seconds since midnight. Because it returns a number, the result can be used in formulas. Please note that the time portion must be in 24 hour clock format.

For example:

```
DTSToSeconds ({table.datetimestring}) + 240
```

This adds 240 seconds (4 minutes) to the DateTime string.

The following is an example used in a record selection formula:

```
DTSToSeconds ({table.datetimestring}) <
30000
```

This record selection formula returns those dates that are before 8:20 am because 30000 seconds after midnight converts to 8 hours, 20 minutes.

DTSToDateTime(DateTime String)

In the DTSToDateTime(DateTime String) function, the DateTime string is converted to actual Date and Time to allow it to be used in comparisons and selection formulas. In Crystal Reports version 8.x, you can use CdateTime() function instead.

For example:

```
DTSToDateTime({table.datetimestring}) + 1
```

This adds a day, so 1996/04/12 08:36:23.00 becomes 1996/04/13 08:36:23.00. If 1.25 is added, the result is plus 1 day and 6 hours

The following is an example of a record selection formula:

```
DTSToDateTime ({table.datetimestring}) <
DTSToDateTime ("1994/12/12 05:21:35.00")
```

This prints records earlier than 5:21:35 AM on December 12, 1994

- Or -

```
DTSToDateTime ({table.datetimestring}) <=
DTSToDateTime ("1994/07/26 05:21:35.00")-7.25
```

This prints records earlier than 11:21:35 PM on July 18, 1994.

DTSToTimeField(DateTime String)

The DTSToTimeField(DateTime String) function converts a DateTime string to an actual time. This allows the result to be used in comparisons and selection formulas:

For example:

```
 DTSToTimeField ({table.datetimestring})-1
// subtracts 1 second from the time
// all comparisons and computations in formulas are based
// on seconds
DTSToTimeField ({table.datetimestring1})-DTSToTimeField
({table.datetimestring2})
// returns a seconds value to add a minute, the formula
// would appear as
DTSToTimeField ({table.datetimestring})+60
// adds 1 minute (60 seconds) to the time
```

The following is an example of a record selection formula:

```
DTSToTimeField ({table.datetimestring}) <
DTSToTimeField ("1994/07/26 08:10:00.00")
// prints records earlier than 8:10 AM regardless of the
// date
```

- Or -

```
DTSToTimeField ({table.datetimestring}) =
DTSToTimeField ("1994/07/26 18:10:00.00")
// prints any records matching 6:10 PM regardless of year
```

DateTime Fields

DateTimeToDate(DateTime)

The DateTimeToDate(DateTime) converts a DateTime field into a Crystal Reports Date value that can be used in a date calculation. In Crystal Reports 8 and higher, use the CDate() formula instead.

For example:

```
DateTimeToDate (DateTime
(1996,08,30,08,31,21))
// returns 1996-08-30
```

NOTE	If you drop the time portion of the DateTime field the time defaults to Midnight (12:00 am) when using the result in calculations.

DateTimeToTime(DateTime)

The DateTimeToTime(DateTime) function converts a DateTime field into a Crystal Reports Time Type value, for use in time calculations

For example:

```
DateTimeToTime (DateTime
(1996,08,30,08,31,21))
// returns 8:31:21 AM
```

DateTimeToSeconds(DateTime)

The DateTimeToSeconds(DateTime) function converts the time portion of a DateTime field into a numeric value that represents the time in seconds.

For example:

```
DateTimeToSeconds(datetime(2001,10,4,3,0,1))
```

Returns (3 * 3600 seconds/hour) + (0 * 60 seconds/minute) + 1 second -> 10,801

DateTime()

The DateTime() function takes date and time parameters and returns a DateTime field that could be used to pass into one of the three DateTime functions as the DateTime parameter. The three DateTime functions are:

- DateTimeToDate(DateTime)
- DateTimeToTime(DateTime)
- DateTimeToSeconds(DateTime)

For example:

```
DateTime(2001,01,30,13,30,15)
DateTime(Date(2001,01,30), Time(13,30,15)]
```

Both return 01/30/2001 1:30:15PM

Downloadable additional DateTime functions

Most of the functions listed in this section will only work in Crystal Reports 5 and higher. In Crystal Reports 8 and higher, there are similar functions that are installed automatically and are available to use without having to download additional functions. In general, the behavior of the functions installed in Crystal Reports 8 and those available for download to use in Crystal Reports 8 is similar.

Crystal Reports does not install the following functions automatically:

- DateTimeDiff(DateTime,DateTime)
- NumberToDate(Number)
- DateToJulian(Date) and JulianToDate(Number)
- Btime(Number)
- PDXTimeToCRTime(number)
- DateTo2000(Date, Number)
- DTSTo2000(DateTime String, Number)
- DateTimeTo2000(DateTime, Number)
- Seconds Since Midnight UFL

If you want to use these functions you will have to either download the appropriate addition function from our website or install them from the Crystal Reports CD.

The following is the link to the Crystal Decisions support site, where you can download the additional function:

http://support.crystaldecisions.com

In the following section, you will find the addition function file name in the corresponding section.

DateTimeDiff(DateTime,DateTime)

Additional function file name: ufldtdif.exe

The DateTimeDiff(DateTime,DateTime) calculates the difference between two DateTime fields in a form that returns. “xxx days xx hours xx minutes xx seconds”.

The following is an example of the syntax this function uses:

```
DateTimeDiff (x, y)
```

Where "x" and "y" are true DateTime fields, not DateTime strings

For example:

```
{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)"

NumberToDate(Number)

For Crystal Reports 7 and 8, you can use the formula in section Converting Fields and Numbers to a Date Type. In Crystal Reports 5 and 6, download the additional function.

Additional function file name: ufltdate.exe

In the NumberToDate(Number) function, the "Number" argument is a NUMERIC value in the format YYYYMMDD. This function will convert it into Crystal Date format.

The following is the format the date will display in after it has been converted:

```
Date(YYYY, MM,DD)
```

For example

```
NumberToDate(19971231)
```

returns Date(1997,12,31) and displays 1997/12/31

DateToJulian(Date) and JulianToDate(Number)

Additional function file name: ufljul.exe

Julian Date fields store numeric values that represent the number of days from a specific starting point. The starting point for JulianToDate and DateToJulian functions is 4713 BC.

The history behind this is that in 1583, the French classics scholar Joseph Justus Scaliger proposed that the epoch of the Julian era be fixed at January 1, 4713 BC.

For example:

```
JulianToDate ({table.datefield})
-or-
DateToJulian (date(1995,12,25))
```

NOTE	Some date fields use a special Julian date format that is based on a particular date, for example January 1, 1900. The following formulas would convert them: Convert the date 12/25/1995 to the special Julian date: **DateToJulian(<date>) -DateToJulian(** **Date(1900,1,1)-1)** If you do not know the starting point of your database's Julian Date, there is a simple method find out. You need to know what the date and the corresponding number. For example, the date is Date (1995,12,25), and the Julian number is 35056. You can create the formula: **(1995,12,25) – 35056;** Place this anywhere on your report and the Julian Date will be displayed. In this example it would display 1900/01/01. You can now just add this date to your field with the following formula code: **Date (1900,01,01) + {Table.JulianDate};** This should now display the correct date in a Crystal Date format.

Btime(Number)

Additional function file name: uflbtime.exe

The Btime(number) function converts a decimal number into a fraction of time based on a 24-hour clock.

For example:

```
btime(decimal number)
```

```
Example
```

```
Btime(.50)
```

Returns 12:00:00

PDXTimeToCRTime(number)

Additional function file name: uflpdxtm.exe

Paradox Time fields are read by Crystal Reports as numbers, representing Seconds after Midnight. This is the way Paradox stores 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 DLL does the conversion automatically, and converts the field into Crystal Reports time format, which allows more flexibility in formatting, and includes support for the Windows time-format settings.

The following is an example of the syntax this function uses:

```
PDXTimeToCRTime({table.timefield})
```

DateTo2000(Date, Number)

Additional function file name: uf52000.exe

The Dateto2000(Date, Number) function evaluates the date value specified and converts it to the 21st century based on the range specified

For example:

```
DateTo2000(97/05/06, 98)
```

returns 2097/05/06

```
DateTo2000(97/05/06, 90)
```

returns 1997/05/06

DTSTo2000(DateTime String, Number)

Additional function file name: uf52000.exe

The DTSTo2000(DateTime String, Number) function evaluates the DateTime value specified and converts it to the 21st century based on the range specified.

For example:

```
DTSTo2000(”05/05/06 12:00:00”,20)
```

returns 2005/05/06 12:00:00 Since 05 is before 20, the last argument of the function, the year is assigned 2005.

```
DTSTo2000(“97/05/06 12:00:00”,20)
```

returns 1997/05/06 12:00:00 Since 97 is after 20, the last argument of the function, the year is assigned 1997.

DateTimeTo2000(DateTime, Number)

The DateTimeTo2000(DateTime, Number) function evaluates the specified DateTime value and converts it to the 21st century based on the range provided.

For example:

```
DateTimeTo2000(DateTime(1998,10,25,8,35,30), 10)
```

Returns 1998/10/25 8:35:30 because 1998 is not before 1910, which is the last argument in the function.

```
DateTimeTo2000(DateTime(“73/12/05
12:00:00”),80)
```

Returns 2073/12/05 12:00:00 because 1973 is before 1980, which is the last argument in the formula.

Seconds Since Midnight Additional Function

Additional function file name: uflssm.exe

There are several functions available in this additional function. The following are a list of three related functions:

- HoursFromSecsSinceMidnight({number})
- MinutesFromSecsSinceMidnight({number})
- SecondsFromSecsSinceMidnight({number})

For example:

These functions convert time fields that are stored as seconds past midnight to standard output formats. This function returns the number of whole hours from numeric value representing the number of seconds since midnight.

For example:

```
HoursFromSecsSinceMidnight(3675)
```

Returns 1. Note that the decimal portion is not returned.

```
MinutesFromSecsSinceMidnight(3800)
```

Returns 3. It returns the number of whole minutes remaining after removing the hour portion.

```
SecondsFromSecsSinceMidnight(80)
```

Returns 20. It returns the remaining number of seconds after removing the hours and minutes.

The following functions convert the number of seconds since midnight into various time formats that are returned as strings:

- HHMMPMFromSecsSinceMidnight({number})
- HHMMSSPMFromSecsSinceMidnight({number})
- HHMMFromSecsSinceMidnight({number})
- HHMMSSFromSecsSinceMidnight({number})

These functions are available in the additional function called uflssm.exe

For example:

```
HHMMPMFromSecsSinceMidnight(3601)
```

returns "1:00 AM"

```
HHMMSSPMFromSecsSinceMidnight(3601)
```

returns "1:00:01 AM"

```
HHMMFromSecsSinceMidnight(3601)
```

returns "01:00"

```
HHMMSSFromSecsSinceMidnight(3601)
```

returns "01:00:01"

The following function converts the seconds since midnight into a Crystal time field:

```
CRTimeFromSecsSinceMidnight({number})
```

WeekOfYear({date field})

Additional function file name: uflssm.exe

The WeekOfYear({datefield}) function returns a number representing the week of the year using Sunday as the first day of the week.

For example

```
WeekofYear(date(2001,1,1))
```

returns 1

```
WeekOfYear(date(2001,1,7))
```

returns 2 because Sunday is the first day of the second week.

NOTE	In Crystal Reports version 8, the Datepart() function is available to return the week of the year. It can take a third argument to specify the day of the week to start on. DatePart("ww",date(2001,1,7)) returns 2 because Sunday is the default starting day. DatePart("ww",date(2001,1,7),2) returns 1 because the third argument defines the starting day as Monday.

Converting fields & numbers to date type in Crystal Reports 5, 6, 7, and 8.x

Strings Dates

Some date fields are stored in a database as a string format that CDate() or DtsToDate() cannot convert. In such cases, it is possible to extract the year, month and day numbers by parsing out those numbers from the string. For example, assume the format 01.31.1998.

The following two methods convert a string date to a Crystal date format:

Method 1

```
Numbervar mm := 0;
NumberVar dd := 0;
NumberVar yy := 0;
//check that the month portion is numeric…
If NumericText({Table.Date}[1 to 2])
Then
//parse out the month portion (first and second characters)
mm := ToNumber({Table.Date}[1 to 2]);
If NumericText({Table.Date}[4 to 5])
Then
//parse out the day portion (fourth and fifth characters)
dd := ToNumber({Table.Date}[4 to 5]);
If NumericText({Table.Date}[7 to 10])
Then
//parse out the year portion (seventh to tenth characters)
yy := ToNumber({Table.Date}[7 to 10]) ;If (mm > 0) and
(dd > 0) and
(yy > 0)
then
Date(yy,mm,dd) else Date(0,0,0);
```

Method 2

If the date field is stored as a string in the format “31 December 1999 11:59:59 PM”. CDate() will convert this correctly but DTSToDateTime() will not. In order to convert this in Version 7 of Crystal Reports, you could use the following formula.

```
 //@convert

//Declare several variables that will be used in the
//formula. The string will need to be broken down into the
//different components of the string

Numbervar DDay;

Numbervar MMonth;

Numbervar YYear;

Numbervar HHour;

Numbervar MMinutes;

Numbervar SSeconds;

Stringvar DayorNight;

Stringvar MON := uppercase(({datestring.field})[4 to 6]);


//Next, a condition needs to be created for each possible
//month, assigning the appropriate numeric value for each
//month of the year.

if MON = "JAN" then

MMonth := 1 else

if MON = "FEB" then

MMonth := 2 else

if MON = "MAR" then

MMonth := 3 else

if MON = "APR" then

MMonth := 4 else

if MON = "MAY" then

MMonth := 5 else

if MON = "JUN" then

MMonth := 6 else

if MON = "JUL" then

MMonth := 7 else

if MON = "AUG" then

MMonth := 8 else

if MON = "SEP" then

MMonth := 9 else

if MON uppercase({datestring.field })[4 to 6] = "OCT" then

MMonth := 10 else

if MON = "NOV" then
```

```
MMonth := 11 else

if MON = "DEC" then

MMonth := 12;


//This next section assigns values to the rest of the
//variables. Each component of the following code extracts
//the values corresponding to the different elements
//contained within the datestring. For example the DAY
//position and the YEAR position in the string will be
//extracted and stored into appropriate variables.

DDay:=  tonumber (trim({datestring.field }) [1 to 2]);


stringvar YearTimeSegment :=
right(trim(datestring.field),16);

YYear := tonumber(YearTimeSegment[1 to 4]);

HHour := tonumber(YearTimeSegment[6 to 7]);

MMinutes := tonumber(YearTimeSegment[9 to 10]);

SSeconds := tonumber(YearTimeSegment[12 to 13];

DayorNight := YearTimeSegment[15 to 16];


//Next we need to determine whether or not our final //time
value in our datetimestring represents AM or PM. //If the
formula is just left as it is now, the time values //that
are extracted would always represent AM as they are
//stored in a regular 12 hour format. To convert the entire
//string into the proper DateTime format, we need to look
//at the value of our DayOrNight variable to see if either
//AM or PM has been stored in it. If it is PM, then we must
//add a value if 12 to the Hhour variable to get the
//appropriate time. Once that is determined we can finish
//off the calculation part of the formula with the
//following.

If HHour = 12 then

(

If DayOrNight = “AM” then HHour := 0;

)

else

(

if DayOrNight = "PM" then

HHour := HHour + 12;

);

//Finally we need to recreate our datetimestring by using
//the variables as arguments once again in the DateTime()
//function. The DateTime() function returns this formula as
//a date field.

DateTime(YYear,MMonth,DDay,HHour,MMinutes,SSeconds);
```

Numeric Dates

If a date is stored in numeric format, for example 01311998, it is easier to use math to convert it into a date than to use the ToText() function with subscripting. There are a number of ways to accomplish this. Here is one method:

```
NumberVar dd:= ToNumber(ToText({Number.Date},0,"")[1 to
2]);

NumberVar mm:= ToNumber(ToText({Number.Date},0,"")[3 to
4]);

NumberVar yy:= ToNumber(ToText({Number.Date},0,"")[5 to
8]);

Date(yy,mm,dd)
```

NOTE	The additional function NumberToDate() can be used to convert a numeric date if it is stored in the format YYYYMMDD.

Number of Seconds since a Certain Date

The date and time is stored as the number of seconds from midnight Jan 1, 1970 Greenwich Mean Time. In one formula field for the date, use this code:

```
NumberVar TimeChange := -5;

NumberVar SecPerDay := 86400;

NumberVar SecPerHr := 3600;

Truncate(({field}-(TimeChange * SecPerHr))/SecPerDay) +

Date(1970, 1,1);
```

This formula returns the date according to a -5 hr difference for the Eastern Time zone from Greenwich Mean Time. The value for TimeChange would be –6 for Central time, -7 for Mountain Time, and -8 for Pacific Time. In another formula field for the time, use this code:

```
NumberVar TimeChange := -5;

NumberVar SecPerDay := 86400;

NumberVar SecPerHr := 3600;

btime( Remainder(({field}-(TimeChange *

SecPerHr)),SecPerDay)/SecPerDay );
```

This returns a time in military format.

NOTE	This formula does not account for changes between Standard and Daylight savings time. As written, the formula fields work with Standard time only. There is no way to include this variation as the changes do not occur on the same dates each year.

Calculating the difference between two DateTime Fields

In Crystal Reports version 5, 6, 7, and 8.x, you want to calculate the difference between two dates. However, one of the following applies to your situation:

- you do not want to use the additional function, Datediff()
- you need to the average DateTime values

To create a formula that calculates the difference between two DateTime fields to display the result in days, hours, minutes, and seconds you must complete the following steps:

1. Convert all the date-times fields to a common time unit, such as seconds.
2. Perform a mathematical calculation using the common time unit.
3. Convert the result to a common time unit, which in this case is seconds, back to a desired format, such as “x days hh:mm:ss”.

For example:

If you have an “Orderdate” and a “Shipdate” field and you want to calculate the average time between the Order date and the Shipping date.

The following sample formula, calculates the difference between the two dates in seconds:

1. From ‘Insert’, select ‘Field Explorer’. This launches the Data Explorer.
2. Select ‘Formula Fields’, and click the ‘New’ icon.
3. Give the formula a name such as Difference.
4. In the Crystal Reports formula editor, create a formula similar to the following:

```
@Difference

// This formula will calculate the

// difference in seconds between the two fields.

({table.shipdate} - {table.orderdate}) * 86400
```

When you subtract two date-time fields, you get a fractional answer that represents the difference in days. To convert the days difference to seconds, you must multiply the difference by 86400, which is the result of 24 hours * 60 minutes * 60 seconds.

To calculate the average of this difference in “days, hours, minutes, seconds”, create a formula similar to the following:

1. From ‘Insert’, select ‘Field Explorer’. This launches the Data Explorer.
2. Select ‘Formula Fields’, and click the ‘New’ icon.
3. Give the formula a name such as Average.
4. In the Crystal Reports formula editor, create a formula similar to the following:

Method 1:

```
//@Average

//calculate the average number of seconds

numbervar AvgSeconds :=
sum({@Difference})/count({@Difference});

local numbervar Days;

local numbervar Hours;

local numbervar Minutes;

local numbervar Seconds;


//find the number of days by dividing by the seconds in a
//day

Days := truncate(AvgSeconds/86400);

//find the number of hours by subtracting the

//day portion and dividing by the number of

//seconds in an hour

Hours := truncate((AvgSeconds - (Days * 86400))/3600);


Minutes := truncate((AvgSeconds - (Days * 86400) - (Hours *
3600))/60)

Seconds := AvgSeconds - (Days * 86400) - (Hours * 3600) -
(Minutes * 60)

//Concatenate the days, hours, minutes, and seconds into a
//string

totext(Days,0) + " day(s) " + totext(Hours,"00") + ":" +
totext(Minutes,"00") + ":" + totext(Seconds,"00");
```

5. Save and close the formula.
6. Place the formula in the Report Footer of the report.

Method 2:

In Crystal Reports version 5, 6, 7, and 8.x, a function called TimeSerial() is available to use. This function can be used as a shortcut to converting total seconds into hours, minutes and seconds. The TimeSerial() function takes three numeric arguments for hours, minutes and seconds and converts it to a time field. It is similar to the Time() function, except that it allows you to enter in minutes or seconds greater than 59. In order to use it in a second conversion formula, the day portion must be calculated separately. The entire result can then be concatenated in a string.

1. From 'Insert', select 'Field Explorer'. This launches the Data Explorer.
2. Select 'Formula Fields', and click the 'New' icon.
3. Give the formula a name such as Average.

4. In the Crystal Reports formula editor, create a formula similar to the following:

```
//@Average

local numbervar AvgSeconds :=
sum({@Difference})/count({@Difference});

totext(truncate(AvgSeconds/86400),0) + " day(s) " +
totext(timeserial(0,0,AvgSeconds),"HH:mm:ss");
```

NOTE	When you format the TimeSerial portion, it is important to specify capital HH for the hour, so that it will display with the 24 hour clock format instead of AM/PM.

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