A

Appendix A

In This Chapter

These are most of the dBase functions that may be used in various areas within the GoldMine Premium application. As you may or may not remember, the dBase functions are still employed throughout GoldMine Premium even when using the Microsoft SQL backend. In fact I recently asked this question directly to FrontRange:

"dBase functions in Filters, screens etc. will not be changing, correct?"

This is the answer that was returned:

"That's the intent. We are not completely sure if we will be able to compensate for all areas, but we do intend to minimize the impact."

Initial testing with this the GoldMine Premium version 9.0.2.36 shows this statement to be true. Actually, I don't see how they could replace dBase which a simpler language for the end users use.

Given that, I have tried to lay these functions out showing the syntax and some examples of usage. Where possible, I have identified locations where the function may or may not be employed. I supplied some functions that we have been able to use, but that are not necessarily supported by GoldMine.

Character Functions

Syntax: at(<Search for Character>, <Search in Character String>)
rat(<Search for Character>, <Search in Character String>)

Abstract: The **at()** syntax was designed to allow users to find a character set within a character value starting from the left hand side of the character value and looking toward the right. The **at()** syntax returns a numeric value, representing the first occurrence of that character set from the left hand side, that can be used in other syntaxes (see the **substr()** syntax). The **at()** function should be thought of as **lat()** (not a valid expression) or search the string left to right stopping at the first occurance of, while the **rat()** function is just the opposite, or search the string right to left stopping at the first occurance of. The first argument is the character set for which you are looking. The second argument is the text string, or field, that you want to have searched for the character set.

Prerequisite: At what position does the period occur in the string "Donald J. Hunt"?

Example: at(".","Donald J. Hunt") rat(".","Donald J. Hunt")

Returned: **9** as a numeric value **9** as a numeric value

Prerequisite: Contact1.Contact = "Donald J. Hunt "

Example: at(" ", Contact1.Contact) rat(" ", trim(Contact1.Contact))

Returned: **7** as a numeric value **10** as a numeric value

Syntax: alltrim(<Character String>)

Abstract: The **all trim()** syntax will remove any leading and any trailing spaces from a character string. Also see, the **ltrim()** syntax and the **trim()** syntax.

Example: alltrim(" DJH ")

Returned: **DJH** as a character value

Syntax: char(<Extract from String>, <Position>) (GoldMine Reports)

Abstract: The **char**acter syntax takes two arguments. The first argument is the string from which you want to extract the character. The second argument is the position number of the character that you want to extract. Also review the **left()** syntax, the **right()** syntax, and the **substr()** syntax.

Example: char("The quick brown fox jumped over the log", 5)

Returned: **q** as a character value

Prerequisite: Contact2.Comments = "The quick brown fox jumped over the log"

Example: char(Contact2.Comments, 11)

Returned: **b** as a character value

Syntax: ctod(<Character Based Date>)

Abstract: The **c**haracter **to d**ate**()** syntax will convert any character-based date to a real date that can then be evaluated against another date value. This syntax can only be used when the argument being passed is a character type in the form of **"mm/dd/yyyy"**.

Prerequisite: Contact1.Key2 = "6/14/2011 "

Example: ctod(trim(Contact1.Key2)) ctod("06/14/2011")

Returned: **6/14/2011** as a date value **6/14/2011** as a date value

Syntax: **left(<Extract From String>, <Length>)**

Abstract: The **left()** syntax takes two arguments. The first argument is the string from which you want to extract the characters. The second argument is the total number of characters that you want to extract from the left side of the string given in the first argument. Also review the **mid()** syntax, the **right()** syntax, and the **substr()** syntax.

Prerequisite: Contact1.Contact = "Donald J. Hunt "

Example: **left("Donald J. Hunt", 6)** **left(Contact1.Contact, 9)**

Returned: **Donald** as a character value **Donald** J. as a character value

Syntax: **len(<Character String>)**

Abstract: The **len**gth**()** syntax will return a numeric value representing the number of characters in the string that is given as an argument.

Prerequisite: Contact1.Contact = "Donald J. Hunt "

Example: **len("Donald J. Hunt")** **len(trim(Contact1.Contact))**

Returned: **14** as a numeric value **14** as a numeric value

Syntax: **lower(<Character String>)**

Abstract: The **lower()** syntax will convert a character string to all lower case letters. This syntax is often used when comparing two strings that may be dissimilar by first equalizing both sides of the equation. Donald J. Hunt does not equal DONALD J. HUNT, but if both are converted to lower case then donald j. hunt will equal donald j. hunt. See the syntax **upper()**.

Example: **lower("DJH")**

Returned: **djh** as a character value

Example: **lower("Donald J. Hunt") = lower("DONALD J. HUNT")**

Returned: **True** as a boolean value

Syntax: **ltrim(<Character String>)**

Abstract: The **l**eft **trim()** syntax will remove any leading spaces from a character string. Also see, the **alltrim()** syntax and the **trim()** syntax.

Example: **ltrim("   DJH   ")**

Returned: **"DJH   "** (notice that the trailing spaces remain)

Syntax: **ltrimpad(<Character String>, <Length>, <Pad Character>)**

Abstract: The **l**eft **trim pad()** syntax will remove any leading spaces from a character string, and then pad the resulting string with the pad character to make the length of the string equal to the request length.

Example: **ltrimpad("   1.00", 8, "$")**

Returned: **$$$$1.00** as a character value

Syntax: mid(<Character String>, <Start>, <Length (optional)>)

Abstract: The **mid()** syntax takes three arguments. The first argument is the character string from which you want to extract a specified number of characters. The second argument is the starting position from which you want to begin your extraction; everything from that point to the end of the string will be extracted if, the optional, Length argument is not added. The Length parameter describes how many characters from the starting position to extract. Also review the **left()** syntax, the **right()** syntax, and the **substr()** syntax.

Prerequisite: Contact1.Contact = "Donald J. Hunt "

Example:	mid("Donald J. Hunt",11, 4)	mid(trim(Contact1.Contact), 7, 7)
Returned:	**Hunt** as a character value	**J. Hunt** as a character value

Syntax: pad(<Character String>, <Length>, <Pad Character (optional)>, <Mode>)

Abstract: The **pad()** syntax gives you a couple of ways to manipulate a character string. The pad character is optional as it will default to a space. There are 3 pad modes:

0 = right pad
1 = center pad
2 = left pad

Example:	pad("207.00", 8, "$", 0)	pad("207.00", 8, "$", 1)
Returned:	**207.00$$** as a character value	**207.00** as a character value

Example: pad("207.00", 8, "$", 2)

Returned: **$$207.00** as a character value

Syntax: padl(<Character String>, <Length>, <Pad Character>)
padr(<Character String>, <Length>, <Pad Character>)

Abstract: The **pad left()** or pad right syntaxes gives you a way to make a numeric character string a specified number of characters in length. If the string being passed is not the specified number of characters, then these syntaxes will insert the character, that you designate, to bring it to the specified length.

Example:	padl("207",6,"0")	padr("207",6,"0")
Returned:	**000207** as a character value	**207000** as a character value

Prerequisite: Contact2.uMyNumber = 122 (as a numeric value)

Example: padl(ltrim(str(Contact2.uMyNumber)),10,"0")

Returned: **0000000122** as a character value

Syntax:	proper(<Character String>)	
Abstract:	The **proper()** syntax will take any character string between quotations or in a field, and convert the first letter of each word to upper case while the remaining letters of that word are converted to lower case.	
Examples:	proper("DONALD J. HUNT")	proper("donald hunt")
Returned:	**Donald J. Hunt**	**Donald Hunt**
Prerequisite:	Conact1.Key1 = "EMPLOYEE "	
Example:	proper(trim(Contact1->Key1))	
Returned:	**Employee**	

Syntax:	right(<Character String>, <Length>)	
Abstract:	The **right()** syntax takes two arguments. The first argument is the string from which you want to have the characters extracted. The second argument is the number of characters, from the right, which you want to have extracted from the string in the first argument. Also review the **left()** syntax, the **mid()** syntax, and the **substr()** syntax.	
Prerequisite:	Contact1.Contact = "Donald J. Hunt "	
Example:	right("Donald J. Hunt", 4)	right(trim(Contact1.Contact), 4)
Returned:	**Hunt** as a character value	**Hunt** as a character value

Syntax:	space(<Number>)	
Abstract:	The **space()** syntax may be used to blank out a field via the use of the Lookup.ini, or may be used in an expression to added a fixed amount of spaces.	
Example:	space(0)	"DJ"+space(6)+"20030201"
Returned:	**empty field** as a character value	**DJ 20030201** as a character value

Syntax:	stod(<String Date>)	
Abstract:	The **s**tring **to d**ate**()** syntax will convert any character-based date to a real date that can then be evaluated against another date value. This syntax can only be used when the argument being passed is a character type in the form of **"yyyymmdd"**.	
Prerequisite:	Contact1.Key2 = "20110614 "	
Example:	stod("20110614")	stod(trim(Contact1.Key2))
Returned:	**6/14/2011** as a date value	**6/14/2011** as a date value

Syntax: strtran(<Character String>, <Search String>, <Replace String>)

Abstract: The **str**ing **tran**sposition**()** syntax allows you to convert a portion of a character string to another character string. This syntax requires that three parameters be passed, all of which are character-based. The first parameter is the string which is to be searched. The second parameter is the string to look for in the first parameter, and the third parameter is the string which you want to substitute for the second parameter.

Example: strtran("20100614", "2010", "2011")

Returned: **20110614** as a character value

Prerequisite: Contact1.Key2 = "20100614   "

Example: strtran(Contact1.Key2, "2010", "2011")

Returned: **20110614** as a character value

Syntax: substr(<Character String>, <Start>, <Length (optional)>)

Abstract: The **substr**ing**()** syntax allows you to parse out any portion of a character string. The first argument is the character string to be parsed. The second argument is the numeric value of the position at which to begin the parsing, while the third argument represents the numeric value of how many characters to parse. You could use the **at()** syntax or the **rat()** syntax as the starting position or ending position.

Example: substr("Donald J. Hunt", 1, 6)

Returned: **Donald** as a character value

Prerequisite: Contact1.Key1 = "Donald J. Hunt   "

Example: substr(Contact1.Key1, at(" ", Contact1.Key1)+1, 7)

Returned: **J. Hunt** as a character value

Prerequisite: Contact1.Key1 = "Donald J. Hunt   "

Example: substr(Contact1.Key1, rat(" ", trim(Contact1.Key1))+1, 4)

Returned: **Hunt** as a character value

Syntax: text(<Number/Date>) (GoldMine Reports)

Abstract: The **text()** syntax takes one argument. This argument can be either a date or a number. The text syntax returns a character representation of the date/number.

Example: text(Contact1.LastDate)

Returned: **06/14/11** as a character value

Syntax: **trim(<Character String>)**
rtrim(<Character String>)

Abstract: The **trim()** and **r**ight **trim()** syntaxes will remove any trailing spaces from a character string. Also see, the **alltrim()** syntax and the **ltrim()** syntax.

Example:	**trim(" DJH ")**	**rtrim(" DJH ")**
Returned:	**" DJH"** as a character value	**" DJH"** as a character value

Syntax: **upper(<Character String>)**

Abstract: The **upper()** syntax will convert a character string to all upper case letters. This syntax is often used when comparing two strings that may be dissimilar by first equalizing both sides of the equation. Donald J. Hunt does not equal donald j. hunt, but if both are converted to upper case, then DONALD J. HUNT will equal DONALD J. HUNT (see the syntax **lower()**).

Example:	**upper("djh")**	**upper("Donald J. Hunt") = upper("donald j. hunt")**
Returned:	**DJH** as a character	**True** as a boolean value

Syntax: **val(<Character String>)**

Abstract: The **val**ue**()** syntax will convert a character string number to a numeric value.

Example:	**val("10.25")**	**val("10.00")**
Returned:	**10.25** as a numeric value	**10** as a numeric value

Example: **val("10.95")**

Returned: **10.95** as a numeric value

Syntax: **word(<Character String>, <Number>) (GoldMine Reports)**

Abstract: The **word()** syntax takes two arguments. The first argument is the string from which you want to have the word extracted. The second argument is the number of the word that you want to extract. Also review the **left()** syntax, the **right()** syntax, and the **substr()** syntax.

Example: **word("The quick brown fox jumped over the log", 4)**

Returned: **fox** as a character value

Prerequisite: Contact2.Comments = "The quick brown fox jumped over the log   "

Example: **word(Contact2.Comments, 8)**

Returned: **log** as a character value

Numeric Functions

Syntax:	**abs(<Number>) (GoldMine Reports)**	
Abstract:	The **abs**olute**()** syntax takes one numeric argument. This syntax converts a number to its absolute value.	
Example:	**abs(-10.153)**	**abs(-15)**
Returned:	**10.153** as a numeric value	**15** as a numeric value

Syntax:	**ceiling(<Number>)**	
Abstract:	The **ceiling()** syntax returns the nearest integer that is greater than or equal to the numeric value.	
Example:	**ceiling(5.2)**	**ceiling(-4.3)**
Returned:	**6** as a numeric value	**-4** as a numeric value

Syntax:	**floor(<Number>)**	
Abstract:	The **floor()** syntax returns the nearest integer that is less than or equal to the numeric argument supplied.	
Example:	**floor(-10.153)**	**floor(3.25)**
Returned:	**11** as a numeric value	**3** as a numeric value

Syntax:	**max(<Number1>, <Number2>) (GoldMine Reports)**
Abstract:	The **max**imum**()** syntax takes two arguments. The first and second argument must be numeric values, and this syntax returns the larger of the two values. This syntax can be used to compare two numeric fields.
Example:	**max(10, 20)**
Returned:	**20** as a numeric value
Prerequisite:	Contact2.uNumber1 = 20 and Contact2.uNumber2 = 50
Example:	**max(Contact2.uNumber1, Contact2.uNumber2)**
Returned:	**50** as a numeric value

Syntax: **min(<Number1>, <Number2>) (GoldMine Reports)**

Abstract: The **min**imum**()** syntax takes two arguments. The first and second argument must be numeric values, and this syntax returns the smaller of the two values. This syntax can be used to compare two numeric fields.

Example: **min(10, 20)**

Returned: **10** as a numeric value

Prerequisite: Contact2.uNumber1 = 20 and Contact2.uNumber2 = 50

Example: **min(Contact2->uNumber1, Contact2->uNumber2)**

Returned: **20** as a numeric value

Syntax: **round(<Number>, <Decimal Places>) (GoldMine Reports)**

Abstract: The **round()** syntax takes two numeric arguments. The first argument is the number that you want to round. The second argument is the number of decimal places to which you want to round.

Example:	**round(10.153, 2)**	**round(10.153, 1)**
Returned:	**10.15** as a numeric value	**10.2** as a numeric value

Syntax: **str(<Number>, <Length>, <Decimal Places>, <Pad Character>)**

Abstract: The **str**ing**()** function must be passed a numeric argument, and will convert that argument to a character-based string. The first argument is the number to be converted. The second argument is the length of the string to which to convert the number, and the third argument is the number of decimal places to be represented in the returned string. The fourth argument is the pad character which is padded to the right to fill the length request.

Example:	**str(132.50, 6, 2)**	**str(132.50, 10, 2)**
Returned:	**132.50** as a character	**" 132.50"** as a character
Example:	**str(132.50, 5, 1)**	**str(132.50, 10, 2, "$")**
Returned:	**132.5** as a character	**$$$$132.50** as a character

Example: **str(132.50, 3)**

Returned: **133** as a character (5 and higher rounds up)

Date Functions

Syntax:	accdate(Contact1.AccountNo)	
Abstract:	The **acc**ount number **date()** syntax is used to extract the creation date of a contact record from the GoldMine AccountNo field.	
Prerequisite:	AccountNo = 98110237007‘!$6Q.Jam	
Example:	accdate(Contact1.AccountNo)	
Returned:	**11/2/1998** as a date value	

Syntax:	age(<Date>)	
Abstract:	The **age()** syntax can be used only when the argument being passed is a date value. The age syntax returns a number value representing the age from the date given, up to today’s date.	
Example:	age(stod(“19481123”))	age({11/23/1948})
Returned:	**62** as a numeric value	**62** as a numeric value
Prerequisite:	Contact1.Key2 = 11/09/1958	
Example:	age(ctod(Contact1.Key2))	
Returned:	**52** as a numeric value	

Syntax:	date()	
Abstract:	The **date()** syntax returns the current date of your computer system, and it returns it as a date value for comparison against other date values. The date syntax does not accept any arguments.	
Example:	date()	
Returned:	**6/14/2011** as a date value	

Syntax:	day(<Date>)	
Abstract:	The **day()** syntax can be used only when the argument being passed is a date value. The day syntax returns a number value representing the day portion of a date value only. If you convert a character to a date first, you may then use that as your argument. Also see date().	
Example:	day(date())	day(ctod(“01/09/1998”))
Returned:	**25** as a numeric value	**9** as a numeric value
Prerequisite:	Contact1.Key2 = 01/09/1998	
Example:	day(ctod(Contact1.Key2))	
Returned:	**9** as a numeric value	

Syntax:	**dobindays(<Date>)**	
Abstract:	The **d**ate **of b**irth **in days()** syntax takes any date argument and returns the number of days remaining to the month/day of the date argument.	
Example:	**dobindays({11/23/1948})**	**dobindays(ctod("01/09/1998"))**
Returned:	**162** as a numeric value	**162** as a numeric value
Prerequisite:	Contact1.Key2 = 01/09/1998	
Example:	**dobindays(ctod(Contact1.Key2))**	
Returned:	**324** as a numeric value (return values dependant on processing day)	

Syntax:	**dow(<Date>)**	
Abstract:	The **d**ay **o**f **w**eek**()** syntax returns the weekday from a date value as a number, this is a 0 based value. **0** = Sunday **1** = Monday **2** = Tuesday **3** = Wednesday **4** = Thursday **5** = Friday **6** = Saturday	
Example:	**dow(date())**	**dow({01/09/2003})**
Returned:	**3** as a numeric value	4 as a numeric value

Syntax:	**doy(<Date>)**	
Abstract:	The **d**ay **o**f **y**ear**()** syntax returns the number of days from the beginning of the year to the argument month/day value.	
Example:	**doy(date())**	**doy({06/23/2003})**
Returned:	**49** as a numeric value	**173** as a numeric value

Syntax:	**dtoc(<Date>)**	
Abstract:	The **d**ate **to c**haracter**()** syntax can be used only when the argument being passed is a date value. The date to character syntax turns a date value into character value to be used when strings are required. The returned value will always be in the form of mm/dd/yy. Also see date().	
Example:	**dtoc(date())**	**dtoc({01/09/1998})**
Returned:	**02/10/03** as a character value	**01/09/98** as a character value
Prerequisite:	Contact2.uMyDate = {1/9/1998}	
Example:	**dtoc(Contact2.uMyDate)**	
Returned:	**01/09/98** as a character value	

Syntax:	**dtos(<Date>)**	
Abstract:	The **d**ate **to s**tring**()** syntax can be used only when the argument being passed is a date value. The date to string syntax turns a date value into character value to be used when strings are required. The returned value will always be in the form of **yyyymmdd**. Also see **date()**.	
Example:	**dtos(date())**	**dtos({06/15/2011})**
Returned:	**20110615** as a character value	**20110615** as a character
Prerequisite:	Contact2.uMyDate = {06/15/2011}	
Example:	**dtos(Contact2.uMyDate)**	
Returned:	**20110615** as a character value	

Syntax:	**fmttime(<Character Time>)**	
Abstract:	The **f**or**m**a**t time()** syntax will convert a 24 hour clock time value to a 12 hour clock time value.	
Prerequisite:	Contact1.LastTime = 21:09	Contact1.LastTime = 08:37
Example:	**fmttime(Contact1.LastTime)**	**fmttime (Contact1.LastTime)**
Returned:	**9:09p** as a character value	**8:37a** as a character value

Syntax:	**wdate(<Date>, <Format>)**	
Abstract:	The **w**rite **date()** syntax takes two argument. The first is a standard date argument, while, with the second argument, one tells the syntax how the resulting character value should be formatted. There are 4 formats available. An example of each is supplied.	
Example:	**wdate(date(), 0)**	**wdate(date(), 1)**
Returned:	**Jun 15, 11** as a character	**Wed, Jun 15, 11** as a character
Example:	**wdate(date(), 2)**	**wdate(date(), 3)**
Returned:	**Jun 15** as a character	**Wednesday, June 15, 2011**

Syntax:	**weekday(<Date>) (GoldMine Reports)**	
Abstract:	The **weekday()** syntax takes one date argument, and returns the string day of the week for the date argument.	
Example:	**weekday(Sys.Date)**	**weekday(Contact1.LastDate)**
Returned:	**Monday** as a character value	**Tuesday** as a character value

Syntax:	**year(<Date>)**	
Abstract:	The **year()** syntax can be used only when the argument being passed is a date value. The year syntax returns a number value representing the year portion of a date value only. If you convert a character to a date first, you may then use that as your argument. Also see **date()**.	
Example:	**year(date())**	**year({06/15/2011})**
Returned:	**2011** as a numeric value	**2011** as a numeric value
Prerequisite:	Contact1.Key2 = "06/15/2011 "	
Example:	**year(ctod(Contact1.Key2))**	
Returned:	**2011** as a numeric value	

Miscellaneous Functions

Syntax: asc(<Character>)
chr(<ASCII Number>)

Abstract: The **asc**ii**()** syntax returns the ascii number equivalent of the character enclosed in the parenthesis. The **char**acter**()** syntax will return the ascii character that is associated with the number in the argument.

Example:	asc("M")	chr(190)
Returned:	**77** as a numeric value	**3/4** as a character value

Syntax: counter(<Character Var>, <Increment Nbr>, <Start Nbr>, <Action Nbr>)

Abstract: The **counter()** syntax returns a sequence of consecutive numbers, incremented as specified, each time that the syntax is evaluated. The Character Variable must be unique and is stored in the Lookup table as is the last used number. The Start Number is the number at which you wish the counter to begin, and should only be used once or when resetting the counter to a position. When the Action is set to 1 the counter is reset to the Start Number. When the Action is set to 2 the counter is removed from the Lookup table.

Example:	counter("AcctNo", 1)	counter("AcctNo", 1, 1000, 1)
Returned:	**1** as a numeric value	**1000** as a numeric value

Syntax: double(<Any Type>) (GoldMine Reports)

Abstract: The **double()** syntax takes one argument, a numeric, character, or date value, and converts this to a numeric value.

Example: double("10.153") (with report field properties set to 2 decimal places)

Returned: **10.15** as a numeric value

Example: double(7/4) (with report field properties set to 2 decimal places)

Returned: **1.75** as a numeric value

Example: double("6/15/2011") (with report field properties set to 0 decimal places)

Returned: **20110615** as a numeric value

Syntax: html2txt(<Character>)

Abstract: The **html2txt()** syntax takes one character string argument, and formats the string removing all html coding with the appropriate character string in plain text. This syntax is primarily used to convert HTML Calendar/History Notes inf reporting purposes, although it can be used anywhere.

Example: html2txt(ContHist.Notes)

Returned: **This is a test for the HTML2Txt syntax** as plain text value

Syntax: httpstr(<Character>, <Option>)

Abstract: The **httpstr()** syntax takes one character argument, usually a website, and formats the string replacing all non-letter/non-number characters with appropriate http percent codes.

Example: httpstr("http://www.DJHunt.US/Beyond Gold.htm", 1)

Returned: **http%3A%2F%2Fwww.DJHunt.US%2FBeyond%20Gold.htm** as a character value

Example: httpstr(Contact1.AccountNo, 1)

Returned: **A1050268152%29%60%2C%3FYRJai** as a character value

Note

*The immediate if function is your most valuable function in your arsenal of functions. It can be utilized in field level **Record Typing**, in your **Lookup.ini**, and many other locations. Make certain that you understand this function fully.*

Syntax: iif(<Logical Expression>, <True>, <False>)

Abstract: The **i**mmediate **if()** syntax will evaluate the first argument, which must result in a True or False when evaluated. If the first argument is evaluated as True, then the second argument will be returned to the calling statement. A False evaluation will cause the third argument to be returned to the calling statement.

Prerequisite: Contact1.Key1 = DJH

Example: iif(Contact1.Key1 = "DJH", "Top Dog", "Low Man")

Returned: **Top Dog** as a character value

Note

*The immediate if function will only compare the equal size components. Since you only asked for a comparison to **"D"** only the first character of the **Key1** field is looked at, therefore, this is also evaluating to **True**.*

Caveat: iif(Contact1.Key1 = "D", "Top Dog", "Low Man")

Returned: **Top Dog** as a character value

Example: iif(trim(Contact1.State) = "MA", "Massachusetts", "Bad State")

Returned: **True** returns Massachusetts
False returns Bad State

Example: iif(trim(Contact1.State) $ "MA NH VT ME CT", "Y", "N")

Returned: if the State field contains MA, NH, VT, ME or CT then **Y** any other state value then **N**

Syntax: int(<Any Type>)

Abstract: The **int**eger**()** syntax takes one argument, a numeric, character, or date value, and converts this to an integer numeric value.

Example: int(10.153) (Works everywhere in GoldMine)

Returned: **10** as a numeric value

Example: int("123.50") (Only works in GoldMine Reports)

Returned: **123** as a numeric value

Prerequisite: Contact2.uMyDate = 6/15/2011

Example: int(Contact2->uMyDate) (Only works in GoldMine Reports)

Returned: **20110615** as a numeric value

Syntax: **reccount()**

Abstract: The **rec**ord **count()** syntax does not accept any arguments, and will return the number of records contained in your dBase or SQL version of GoldMine as a numeric value. This number will represent all records, including deletions in the dBase version of GoldMine.

Example: **reccount()**

Returned: **16819** as a numeric value

Syntax: **recno()**

Abstract: The **rec**ord **n**(**o**)umber**()** syntax does not accept any arguments, and will return the current table pointer record number as a character value. This syntax will return RecID on SQL databases.

Example: **recno()**

Returned: **121** as a character value

Syntax: **time()**

Abstract: The **time()** syntax returns the current time of your computer system, and it returns it as a date value. The time syntax does not accept any arguments. The time is returned as a 24 hour value in the format of hh:mm:ss

Example:	**time()**	**left(str(time()), 5)**
Returned:	**21:19:40** as a date value	**21:19** as a character value

Syntax: **user2full(<Character UserID>)**

Abstract: The **user 2 full()** syntax will return the full UserID and User Name, as displayed in GoldMine pick lists. The first example shows you the syntax in its natural usage while the second example shows it in combination with other syntaxes to extract the users full name.

Example: **user2full("DJ")**

Returned: **DJ (Donald J. Hunt)** as a character value

Prerequisite: Contact1.Key1 = DJ

Example: **strtran(substr(user2full(Contact1.Key1),
at(" ",user2full(Contact1.Key1))+2,255),")"," ")**

Returned: **Donald J. Hunt** as a character value

Syntax: **UserInGrp(<Integer/String Group>)**

Abstract: The user in group syntax will return a True if the currently logged in user of GoldMine is in the passed user group number or the passed user group name.

Example:	**UserInGrp(2)**	**UserInGrp("Sales")**
Returned:	**.T.** as a logical value	**.T**. as a logical value