LOOKUP.INI Guide with examples

Products / Topics :

GoldMine Premium Edition

Applies to

GoldMine Premium Edition - GoldMine Premium Edition 2017.1, GoldMine Premium Edition 2018.2, GoldMine Premium Edition 2019.1, GoldMine Premium Edition 2021.2

Categories

Administration_

Created Date

Apr 14, 2019 10:13:34 AM

Last Modified Date

Dec 11, 2021 3:06:55 AM

Article Number :

000042541

Details

GoldMine Lookup.ini

Overview

You can define conditions to update GoldMine field data based on data in other fields in a LOOKUP.INI file. You can set GoldMine to update under the following conditions:

- **Automatically when data is entered in one or more specified fields**
- **When triggered as an automated process event**
- **When a new record is created**

Each section in LOOKUP.INI contains instructions for one field that you want GoldMine to update. For example, the [City] section will contain update instructions for the City field. The section can contain three types of instructions:

Lookup instructions

Results of the process and updated values

Optional settings

You can create separate lookup rules for each contact file by placing a LOOKUP.INI file in each contact file directory.

The most common error encountered when dealing with the LOOKUP.INI file is making spelling mistakes -these will cause GoldMine to ignore instructions, as it cannot find the field that is incorrectly referenced.

Uppercase or lowercase text will not affect the functionality of LOOKUP.INI, but proper case is advisable for readability.

Basics of the LOOKUP.INI

A. **LOOKUP.INI does not exist; it must be created using a text editor and is written as a standard Windows 'Configuration' file similar in many respects to the WIN.INI and SYSTEM.INI files. The LOOKUP.INI is saved in the shared GoldMine directory.**

The LOOKUP.INI can perform several functions based on a trigger and a resulting action. These functions are as follows:

● Update a CONTACT1 or CONTACT2 field based on an entry in a CONTACT1 or CONTACT2 field.

● Update a CONTACT1 or CONTACT2 field based on a given expression.

● Launch an external application based on a given expression or the occurrence of a specified field entry.

● Apply an activity color code based on either an activity type, an activity code, or both when an activity is scheduled.

● Update a CONTACT1 or CONTACT2 field, or launch an external application when a new record is created.

The first section of the LOOKUP.INI is the [AutoUpdate] section; this specifies trigger fields (fields that will trigger an update if modified) and the update field (resulting fields to be updated). The second section usually comprises the specific instructions and parameters used when updating the GoldMine field.

Other sections that can also be included are as follows:

- **[OnNewRun] and [OnEditRun] determine the external application**

  that is to be launched when a new record is created (either a contact record or a supplemental file record, i.e., adding a new calendar activity).

- **[CalCIrCode] specifies what color is to be assigned to the activity that is being scheduled.**

- **[OnNewRecord] specifies what color is to be assigned to the activity that is being scheduled.**

Updating CONTACT1 and CONTACT2 Fields when creating new contact records

REQUIREMENT 1*:* A company requires that the Salesperson field (USALESREP) be automatically updated [AutoUpdate] when the GoldMine user creating a new contact record. GoldMine can be configured to update one or more fields automatically when a user creates a contact record by including the following statement in the [AutoUpdate] section in LOOKUP.INI NewRecord=field(s)

[AutoUpdate]

NewRecord=UsalesRep

[USalesRep]

LOOKUP1=Contact1->City

London=Brian Chigwell

Birmingham=Colin Wainthrop

Manchester=Samuel Smith

Updating CONTACT1 and CONTACT2 Fields on existing contact records

REQUIREMENT 2*:* A company requires that the Salesperson field (USALESREP) be automatically updated [AutoUpdate] when the GoldMine user is changing the data in the City field (CITY).

The first section of the LOOKUP.INI is headed [AutoUpdate] and is where the trigger and update fields are defined in the format: Trigger field=Update field(, Update field2, Update field3,etc.)

There is no need to prefix field names with a database (CONTACT1->CITY) because no two fields in the CONTACT1 or CONTACT2 data files (except ACCOUNTNO) have the same field name.

When the trigger field (City) is updated, the LOOKUP.INI will evaluate the subroutine of the same update field name (USALESREP). The result of this evaluation determines what is entered into the USALESREP field.

In the example below, the subroutine [USALESREP] contains a reference to the trigger field (City) and a list of potential values that could be contained within the City field (London, Birmingham, Manchester).

If a match is found, for example, the CITY field contains Manchester, and the LOOKUP.INI will populate the USALESREP field with Samuel Smith (Manchester=Samuel Smith).

For every possible entry in the CITY field, a Lookup should be listed if it is to update another field.

[AutoUpdate]

City=UsalesRep

[USalesRep]

LOOKUP1=Contact1->City

London=Brian Chigwell

Birmingham=Colin Wainthrop

Manchester=Samuel Smith

ADDITIONAL NOTE:

While there might be only exceptional cases where such a logic might be applied, it is not necessary that the TRIGGER field must be also the LOOKUP field. The following construct would be possible where the Trigger field is the CITY field, but the LOOKUP field might not be the CITY field but instead the COUNTRY field

[AutoUpdate]

City=UsalesRep

[USalesRep]

LOOKUP1=Contact1->Country

France=Brian Chigwell

United Kingdom=Colin Wainthrop

Germany=Samuel Smith

Check a Sequence of Fields in Order to Update a Field

REQUIREMENT 3: A company requires that the Salesperson field (USALESREP) be automatically updated when the GoldMine user is changing data in the City field (CITY). If no city is matched, the County (STATE) field must be evaluated so that a second attempt is made at populating the Salesperson field.

This example is similar to Requirement1; however, it is now necessary to apply a second LookUp Command. If GoldMine cannot match the CITY field with one of the listed values, it will move on to the next lookup. If required, we can define a maximum of nine different lookups.

If the City field does not contain either London, Birmingham, or Manchester, then the County (STATE) field will be evaluated. If the County returns a value of Berkshire, then the UsalesRep field will be populated with Brian Chigwell.

The LOOKUPx instructions are processed sequentially, which means if the CITY contains a matching value, for example, Birmingham any further values or LOOKUPx instructions are NOT validated anymore. In this case, it would be regardless if the STATE would be Berkshire

Again, it is recommended that every possible or necessary parameter should be set if it is to populate the field correctly

[AutoUpdate]

City=UsalesRep

[USalesRep]

LOOKUP1=Contact1->City

London=Brian Chigwell

Birmingham=Colin Wainthrop

Manchester=Samuel Smith

LOOKUP2=Contact1->STATE

Berkshire=Brian Chigwell

Cheshire=Samuel Smith

Derbyshire=Colin Wainthrop

If No Match is Found

REQUIREMENT 4: A company requires that the Salesperson field (USALESREP) be automatically updated [AutoUpdate] when the GoldMine user entering data populates the City field (CITY). If no county is entered or if an erroneous entry is made, the Salesperson field is to be updated with Unallocated.

The LOOKUP.INI can be set to update a field with a set entry if no matches are found. The "OTHERWISE" statement is included after all LOOKUPs 1-9 have been listed.

In the example below, if the CITY field does not contain London, Birmingham, or Manchester, USALESREP will be populated with Unallocated.

[AutoUpdate]

City=UsalesRep

[USalesRep]

LOOKUP1=Contact1->City

London=Brian Chigwell

Birmingham=Colin Wainthrop

Manchester=Samuel Smith

Otherwise=Unallocated

If the Field to be Updated is Pre-Populated?

GoldMine will not automatically overwrite an existing value in a field to be updated unless it specified to do so. The OVERWRITE statement is a toggle statement (OnIOff or TruelFalse).

- **If Overwrite=0 then existing values in update field are not overwritten.**

  Update the previous example LOOKUP.INI to reflect the new statement.

- **If Overwrite=1 then existing values in update field are overwritten.**

The statement is placed at the end of the subroutine.

[AutoUpdate]

City=UsalesRep

[USalesRep]

LOOKUP1=Contact1->City

London=Brian Chigwell

Otherwise=Unallocated

Overwrite=1

Update Two Fields from One Trigger Field

REQUIREMENT 5: A company requires that the Salesperson field (USALESREP) and the Region field (UREGION) both be populated when the City (CITY) field is populated.

This works in the same way where just one field is being populated. The trigger field is to launch two subroutines rather than one. Note the AutoUpdate section containing a reference to both fields, separated by a comma, and the existence of two separate subroutines.

The order of reference is only important if one field must be updated first because its new value will affect the entry in the second field.

If updating both the City and the Salesperson field were to affect the update of the Region, then the USALESREP section must be referenced first under the AutoUpdate section.

If updating the Region and the City were to affect the Salesperson, then the UREGION field must be referenced first in the AutoUpdate section. If the fields are to be updated independently, then either can be referenced first.

[AutoUpdate]

City=UsalesRep,Uregion

[USalesRep]

LOOKUP1=Contact1->City

....

[URegion]

LOOKUP1=Contact1->City

....

Update a Field Based on an Expression

The LOOKUP.INI, by way of expression, returns an actual value to a field that can be used like every other field within GoldMine.

The AutoUpdate section is identical to the examples described above. It contains a trigger field and a reference to a subsection that is to be actioned.

The subroutine is to contain an expression, rather than a list of returned values from a Lookup1. However, the subroutine must be written in a standard way, and therefore

a dummy Lookup1 must be inserted - although it will be set to not return any values so that the Otherwise command is invoked.

The expression is to be used with the Otherwise command but must be prefixed with an ampersand (&) so that the LOOKUP.INI evaluates the expression as a xBase command, rather than text that is to be inserted in the field.

[AutoUpdate]

UsalesQtr1=UTotal

UsalesQtr2=UTotal

UsalesQtr3=UTotal

UsalesQtr4=UTotal

[UTotal]

Lookup1=Contact1->Company

Otherwise=&(Contact2->UsalesQtr1+Contact2->UsalesQtr2+Contact2->UsalesQtr3+

Contact2->USalesQtr4)

Overwrite=1

The Calculation of Numeric Fields

REQUIREMENT 6: A Company has quarterly sales figures (USALESQTR1-USALESQTR4) for each customer. They require a Total (UTOTAL) field that sums quarters one to four and returns a value.

The AutoUpdate section includes four trigger fields: UsalesQtr(1-4). When any one of these fields is updated, the LOOKUP.INI will run subroutine UTotal.

The Lookup1 command looks in a dummy field. In the example, Contact1->Company is referenced. No values are listed under LOOKUP1, so therefore no match is found. The LOOKUP.INI progresses to the Otherwise command, where the xBase expression is stored.

```
[AutoUpdate]

UsalesQtr1=UTotal

UsalesQtr2=UTotal

UsalesQtr3=UTotal

UsalesQtr4=UTotal


[UTotal]

Lookup1=Contact1->Company

Otherwise=&(Contact2->UsalesQtr1+Contact2->UsalesQtr2+

Contact2->UsalesQtr3+Contact2->USalesQtr4)

Overwrite=1
```

Calculation Fields Using Decimal Places

If the LOOKUP.INI is to calculate a value that includes a decimal place, then each field that is referenced when calculating must also contain the same amount of decimal places, whether needed or not.

The LOOKUP.INI is written in exactly the same way, except that the Otherwise command needs to include a String statement (STR) in order to calculate to x decimal places.

[AutoUpdate]

UsalesQtr1=UTotal

UsalesQtr2=UTotal

UsalesQtr3=UTotal

UsalesQtr4=UTotal

[UTotal]

Lookupl=Contact1->Company

Otherwise=STR(Contact2->UsalesQtrl+Contact2->UsalesQtr2+Contact2->UsalesQtr3+

Contact2->UsalesQtr4,2,10)

Overwrite=1

In the example above, the Otherwise statement ends in 2,10 where 2 indicates the number of decimal places and 10 the length of the field.

Averaging Blank and Filled Fields Using Hypothesis Values

If the Otherwise command held an expression that calculated the sum of the contents of all of the fields then divided the total by the number of fields, those fields that are empty (set to zero) would also be included, i.e.,

Otherwise=&((Contact2>Ufield1+Contact2->Ufield2+Contact2->Ufield3)/3)

If the field values were 10, 20 and 30, the LOOKUP.INI would return a value of 20 (60 divided by 3).

If the field values were 10, 20 and 0, the LOOKUP.INI would return a value of 10 (30 divided by 3).

This inclusion of the blank or zero fields may be inappropriate. That is, in our example the final calculation should be divided by 2, not 3, resulting in an average of 15.

To calculate only the filled fields, the expression must include Counter fields. To do this, each field must be evaluated to see if it is true or false, empty or full before the final calculation can take place. See the example below.

TruelFalse fields are calculated using an if statement, followed by the value that would indicate whether the value is true, then suffixed with a value to replace if true, and then a value to replace if false, i.e., (iif(field=value,true[x]false[y])

In the example above, the expression otherwise&(iif(contact2->ufield1=0,0,1)) states that the value in Ufield1 should equal 0; if it does (true), the Counter field will be populated with 0. If the field is not equal to 0 (that is, it contains either a higher or lower value), then the counter field will be populated with 1 for false (not true). Each field to be evaluated must have its own counter field.

The UTotal section then calculates the sum of all of the counter fields and uses that figure to divide the total of the fields Ufield1-UfieId3.

NOTE: The programming standard of 0 = false and 1 = true has no bearing in this example.

[Autoupdate]

ufield1=ucount1, utotal

ufield2=ucount2, utotal

ufield3=ucount3, utotal

utotal=utotal

[ucount1]

Lookup1=contact1->company

otherwise=&(iif(contact2->ufield1=0,0,1))

Overwrite=1

[ucount2]

Lookup1=contact1->company

otherwise=&(iif(contact2->ufield2=0,0,1))

Overwrite=1

[ucount3]

Lookup1=contact1->company

otherwise=&(iif(contact2->ufield3=0,0,1))

Overwrite=1

[UTOTAL]

Lockup1=contact1->company

otherwise=&((contact2->ufield1+contact2->ufield2+contact2->ufield3)/

(contact2->ucount1+contact2->ucount2+contact2->ucount3))

Overwrite=1

Launching External Applications

You can automatically launch external applications when new records are added and edited in GoldMine, so further processing could be done outside of GoldMine as soon as data in GoldMine is updated.

The [OnNewRun] allows you to set up external applications for each type of contact related record, while the [OnEditRun] launches applications when a record is changed.

Records in the following files can be considered by the [OnNewRun] and [OnEditRun] sections: Contact1, Contact2, Cal, ContHist and ContSupp. In addition, specific record types in Cal, ContHist and ContSupp can be set up to launch different applications.

A specific example would be launching a post-code application whenever the post-code is updated.

To specify the database on creation, simply enter the database name = to the external application.

To specify a record type (RECTYPE) within the database, simply suffix the database name with -RECTYPE, e.g., Contsupp-P or Cal-A.

SECTION	DESCRIPTION
[OnNewRun]	
Contact1 =C:\MYAPP.EXE	When a Contact1 record is created, execute MYAPP.EXE

ContSupp-P= C:\MYPROF.EXE When a profile record is created, execute MYPROF.EXE

Otherwise=C:\ANYAPP.EXE Upon creating any new record that is not specified above, run the external program ANYAPP.EXE. When this program is executed it is suffixed with the File + RECTYPE, i.e., MYAPREXE CAL-A if a Calendar appointment was scheduled.

AppendRecNo=1 Appends the record number to the executable, e.g., the application to execute is MYAPP.EXE 123 where 123 is the record number.

DisableFromAP=1 If an automated process creates the record, then do NOT execute the program.

[AutoUpdate]

Key3=Key5

Zip=Zip

[Zip]

LOOKUP1=Zip

RUN=c:\GoldMine3\GADDR32N.EXE

RUNFLAGS=2

This lookup is created when the Quick Address software is installed with GoldMine. If LOOKUP.INI already exists, the changes will be appended to the existing file.

The Runflags statement when used with [OnEditRun] determines under what circumstances the application should be launched.

SECTION DESCRIPTION

[OnEditRun]

Contact1=C:\MYAPREXE When a Contact1 record is created, Execute MYAPREXE

ContSupp-P=C:\MYPROF. EXE When a profile record is created,

execute MYPROF. EXE

Runflags

The Runflags statement can be placed after the Overwrite statement to inform GoldMine when to run an external application.

Runflags=1 Only when the field's lookup value is found.

Runflags=2 When the field is updated via [AutoUpdate].

Runflags=4 When the field is updated via Automated Process.

Combinations of the above can be made and will work on an AND basis, that is, both conditions must be true. For example:

Runflags=3 When a Lookup value is found and when the field is updated via [AutoUpdate].

Runflags=6 When the field is updated via [AutoUpdate] and when the field is updated via an Automated Process.

Calendar Color Codes

A. **LOOKUP.INI can automatically assign colors to specific activity types (such as calls and appointments) and activity codes. The Activity type (RECTYPE) must be specified with the specific activity code (optional) and the color to be assigned. These details are stored under a new section in LOOKUP.INI called [CalClrCode].**

The RECTYPE field is held in the Calendar and History databases and defines the type of activity record held in those databases. The different Rectypes are detailed below:

RECTYPE	DESCRIPTION
A	Appointment
T	Next Action
M	Message
O	Other
C	Call Back
D	To-Do
S	Forecasted Sale
E	Event

Colors may be assigned using the values in the table below:

CODE	COLOR
0	Bright Blue
1	Bright Purple
2	Bright Red
3	Bright Cyan
4	Bright Green
5	Bright Yellow
6	Cyan
7	White
8	Grey
9	Red
10	Green

11 **Yellow**

12 **Blue**

13 **Purple**

14 **Dark Grey**

15 **Black**

In this example:

Appointments will be bright green (A=4).

Calls will be bright purple *(C1).*

Appointments with an activity code of HOT will be bright yellow (A-H0T5).

Calls with an activity code of CCA will be bright red (C-CCA=2).

[CalClrCode]

A-HOT=5

C-CCA=2

A=4

C=I