4

In This Chapter

The ability to add user defined fields affords the GoldMine Administrator the ability to add fields to their GoldMine application to meet the needs for their particular organization. A field, as I define it, is a place to store one piece of information. For example, the **Source** field, a default field in GoldMine, allows one to store the original single source through which a particular contact became familiar to your organization. This field is character based, and it is defined to be 20 characters in length. If the contact first came to know of the organization through a trade show, and then later reestablished contact with the organization through a marketing campaign, the Source field would not be the place to store and analyze this type of information. An organization that desired to analyze and track multiple contacts would be better served by using the GoldMine one-to-many capability of the **Details** tab. However, I would be heading off on a tangent with that one, and should follow the straight and narrow discussing **Custom Fields**, **Custom Screens**, **Screen Design** and **Record Typing**.

I have combined these items into this single chapter as they go together, hand-in-hand. One must create Custom Screens in order to present the Custom Fields to the users for data entry. Once created, they could be utilized by the Record Typing feature of GoldMine. In rare circumstances, one may want to create user defined fields, and not place those fields on a screen for data entry or data editing. Sometimes this type of user defined field is auto-populated through use of the **Lookup.ini**, discussed in detail later on in another chapter. Such a field could be a counter field that is auto-populated with a unique number. Most often, however, user defined fields are placed on user defined screens, and if they must have special characteristics, those are handled on either the screen level or the field level. I'll discuss this as I talk about each item individually.

Custom Fields

WARNING

Prior to doing any customizations, you are warned to backup your GoldMine databases. Your only recover from an errant deletion is via a restoration of the database.

You have been warned.

We will now proceed with **Step 1** of my process, that of creating the custom fields. You must create your custom fields, then create your custom screens or visa versa, and finally, meld the two together to create a unique custom display for your information. We can now proceed to Custom Fields, and it is here where we collide with the new learning curve inherent with change. The new location for this GoldMine menu item is:

Tools
Configure ►
Custom Fields...

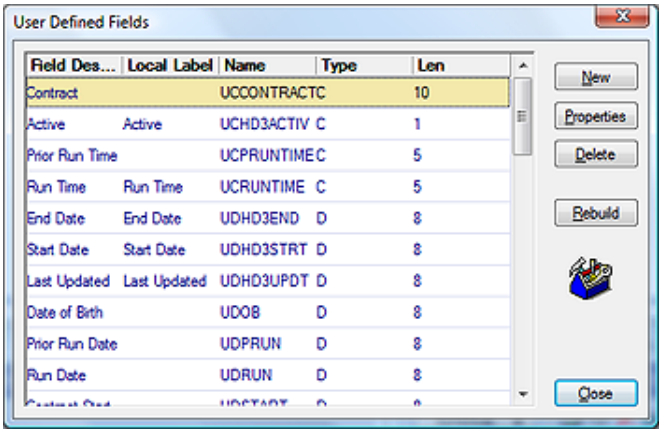

Figure 4-1

This will bring up the dialog form shown here in Figure 4-1. One of the first things that the astute reader will notice, if theirs is a new installation of GoldMine, is that there are 16, yes count them, 16 user defined fields in the GoldMine default configuration. FrontRange has taken the liberty of predefining some of your user defined fields for you. Immediately, I would **Delete USERDEF11** through **USERDEF16** inclusive as they are not so very user

defined now, are they? As for the other 10 user defined fields, which were not defined by any of your users, I usually take them right out of play by reducing their length to 1 character, and removing their description. Oh, and in case you were wanting to give it a try, the deletion of USERDEF01 through USERDEF10 is forboden. I find that they are not anything that I would define, and that they do not follow any convention that I would wish to employ. We'll talk about conventions in a little bit, but first let's eliminate some of those buttons that were shown in the Figure 4-1 dialog form.

The **Delete** button does just that which is stated (go figure), it deletes the selected (highlighted) user defined field from the **ContUDef** list which, when the **Rebuild** takes place, will in turn remove it from the **Contact2** table.

The **Rebuild** button, on the other hand, plays a much larger roll. After one has added, changed the properties of, or deleted fields from the **User Defined Fields**, Figure 4-1, dialog form, GoldMine will ask them to rebuild the tables. This assumes that the user forgot to click on the **Rebuild** button themselves, and has attempted to **Close** the dialog form. Either way, it takes the clicking of this button to add, modify, or remove the definition record from the **ContUDef** table. This is the table where all of the definitions for user defined fields are maintained, the user defined data dictionary if you will. Once this dictionary has been modified accordingly, then GoldMine can proceed to modify the related structure of the **Contact2** table based upon the definitions contained in the **ContUDef** table. Each record in the **Contact2** table will need to have fields added, modified, or removed from them accordingly. Tables, indices and structures will be presented in more detail later in this book.

Note

In the past, I used to warn individuals that they could not add more than 230 user defined fields through this GoldMine dialog form for the GoldMine Premium. Now, although I haven't actually tested it personally, others have, and I am told that you can now add as many fields as you would like here. As I mentioned in the past, this was a BDE limitation, and in GoldMine Premium, with the switch to ADO, this limitation has gone away although there is still a SQL Database limitation of 8,196 bytes per record in any table.

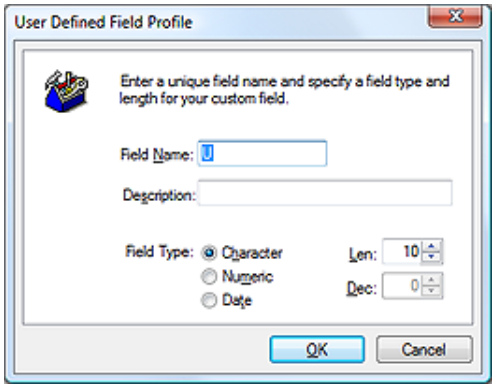

Figure 4-2

The **Properties** button will bring up a dialog form, similar to that shown here in Figure 4-2, however, unlike the **New** button dialog form, which I am showing you in Figure 4-2, all of the information will be contained there on, and it will be ready for any modifications.

Lastly, I will discuss this dialog form as presented to the user when they click on the **New** button. The user will notice that there are a very limited number of properties that they may ascribe to each field. This has always been an issue for GoldMine users.

The **Field Name:** automatically contains a **U** as the first character in the field. All user defined fields in GoldMine must begin with the letter U. If you overwrite the preassigned letter U, then when you click on the **OK** button, GoldMine will append one to the beginning of the field name for you. User defined field names, in GoldMine, may only be 10 characters in length, and the first character must be U. They may not begin with a special character or a number. Based on that, the user has 9 alphanumeric characters with which they may name their user defined field. I always try to assign some relevant meaning to this name, i.e. a field that is to contain a date of birth might be named, udDOB. This field name is apt to appear on a GoldMine field list at some point in time as you are using GoldMine. This convention (Hungarian convention) lets one immediately know that it is a user defined field (**u**dDOB), that it is a date based field (u**d**DOB), and that its name is ud**DOB**. Would one have a better understanding of what might be contained in a field from that list, if one saw UserDef08, or udDOB? That was a rhetorical question, naturally. Obviously. I don't care for the GoldMine field naming convention, and I suggest that you develop and employ a field naming convention that is meaningful to both you and your end users alike. Your only limits are the 9 character length of the field name, and that you may not employ spaces and/or special characters in the name.

The **Description:** field has a little more significance in GoldMine Premium. When one places the field onto a screen for user data entry, GoldMine will take the first 15 characters of the description as its best guess for its **Global Label**. Before this field is placed on a screen for data entry, in some cases, GoldMine will display the description instead of the field name on that list. I just try to make the description as meaningful as I can within the 25 character **Description:** field limit.

Note

*Alas, no changes in GoldMine Premium. There are still no **BLOb** (**B**inary **L**arge **Ob**ject) user defined field type. It is this field type that one would need to create a user defined **Notes** type of field. This has often been requested by end users, and may find its way into future builds of GoldMine. Additionally, check boxes and radio buttons, which are **Logical** (Boolean) type fields, are not available at this time.*

Ah yes! **Field Type:**, a radio button selection for one of three possible data types:

- ◉ **Character**
- ○ **Numeric**
- ○ **Date**

That's it, there aren't any other types, at least not as of this writing. There have been a multitude of requests for memo field types, or logical field types. I want to say, however, that things that are often requested, usually make it into future releases of GoldMine (of course this is the 7th rendition of this book in which I have made this very same statement). If having additional field types is an issue

for you, and it is for most of us, then you are requested to send your suggestions to: **Suggestions@FrontRange.com**. The more requests they receive, the more likely that we'll see these incorporated into a future release of GoldMine.

Let's examine the default ⦿ **Character** data type of field. This type of field can hold any type of character, number, or special characters. A common mistake is to use a numeric field to hold number characters. When a field will hold phone numbers, or social security numbers, the field should be a character based field. Should you want the field to display currency, $1,345.67, you would need to make the field a character based field. Once you have decided that a particular field is to be character based, you must then designate how many characters the field should be allowed to hold. You would do this, setting the limit, using the **Len:** field. The spinner control associated with the **Len:** field will allow you to spin up or down with the default character length of **10** characters.

One could also choose to define a field as a O **Numeric** data type field. These type of fields would, most probably, have mathematical operations performed against them. If you were to set your field to this field data type, you would also be required to define the **Len:** of the field, as well as the number of **Dec:** (decimal places) that the field should carry. When you consider the length for a numeric field, remember to add on an additional place for the sign of the field plus (**+**) or minus (**-**). You must also consider that the decimal itself, if you are having one, will take one space. A number like 1234.67 would require a length of 8 with 2 decimal places. Don't forget Murphy's Law either, try to allow for the possibility that you may have a larger number to contend with in the field in the future.

The last field data type is the O **Date** field type. Should you select the field to carry a date value, then the length will be set for you automatically. There are no other settings about which you should be concerned.

This, then, just about does it for **Custom Fields**. There is just not that much more that I can discuss on this issue. Add as many fields as you desire within the previously noted limitation. You can modify them at any time by highlighting them, and then by clicking on the **Properties** button. The same dialog as is shown in Figure 4-2, will be brought up with the current field definition, and you may edit the settings there.

WARNING

*As of this writing, GoldMine Premium does not display all numeric values correctly. For instance: **1.00** will be displayed as **1**. Whereas, **1.90** would display as **1.9**. Trailing 0's are truncated for some reason that elludes me.*

If you need Numeric fields, you are advised to mirror the Numeric field with a displayed Character based field for data entry, and to then have the Lookup.ini (refer to Chapter 6) populate the Numeric field while properly formatting the displayed Character field.

Custom Screens

I now take up the discussion of **Custom Screens**, please do not confuse this with **Screen Design** as will be discussed later in this chapter. **Custom Screens** are user defined screens for displaying **Contact1/Contact2** fields and/or **Expression** fields under the **Fields** tab and/or on their own designated tabs as the designer so chooses. Think of these as blank pieces of paper which one will later cut holes into to expose certain data fields to the end user for data viewing/entry/modification.

Tip

*Users not possessing Master Rights will still see **Custom Screens...** and **Custom Fields...** visible on their menu even though these selections will not function for these users. The GoldMine Administrator would be advised to remove these selections from a non Master Rights user menu.*

One must be a user possessing Master Rights in order to select

Tools
Configure ►
Custom Screens...

from the GoldMine menu. A user possessing Master Rights will also be able to select Screens Setup... from the local menu when they right-click in the screen area under the Fields tab. Either of these selections will bring the user to the **Custom Screens Setup** dialog shown here in Figure 4-3, although, for a new GoldMine Premium installation the default screens will cause the **Custom Screens Setup** screen to appear different than that which is displayed here. It is from this dialog form that the user has the ability to change the **Properties** of an existing screen, add or **Clone** a **New** screen or **Delete** an existing screen. Each of these methods has a button except for the **Delete** method. The **Delete** method must be accessed through the **Local Menu** (right-click in the area under the heading row), or via the keyboard, **Delete** key, and will act upon the highlighted record on this dialog form. As I state in the sidebar **Note**, you will not be able to delete the currently active screen.

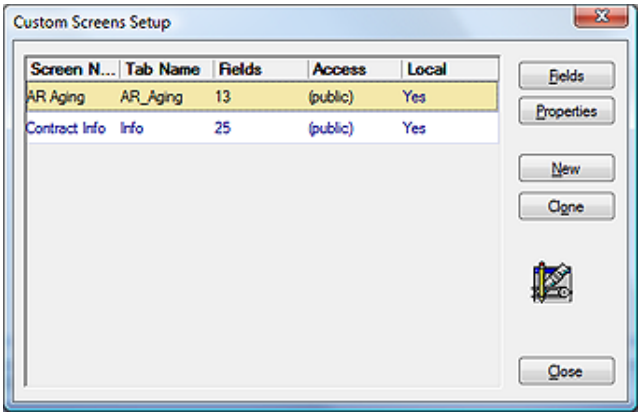

Figure 4-3

Note

GoldMine will not allow the active screen to be removed. There must always be at least one user defined screen even if you don't plan on using the screen for your organization.

Tip

*This Tip may or may not hold true in GoldMine Premium 8.5.1.12, however, remains a good tip just the same. Screens on the GoldMine local menu, under the **Fields** tab, will appear in the order in which they are created, and will not be in alphabetical order, for example. I always create my screens without regard to order, and, when finished, I clone them in the order that I want them to appear on the local menu list. After I am through with my cloning process, I simply remove those screens from which the clones were derived.*

GoldMine Premium now has five screens created in the default installation, the **End User** screen, the **Car** screen, the **Service Customer** screen, the **Other** screen, and the **Prospect** screen. None of which are particularly useful to most organizations, and most will delete them, save one, which they

Note

GoldMine will permit the addition of 19 user defined screens for a total of 20 user defined screens. Of those, only up to 18 may be defined to have their own tab. The 18 user definable tabs must be shared between user defined Screen tabs, and user defined Detail tabs.

will modify. I had started by highlighting the **Car** screen, right-clicking on it, and selecting **Delete...** from the local menu to remove it. I then continued the process until all of the screens had been removed except for the **End User** screen. I next highlighted the **End User** screen, and I clicked on the **Fields** button.

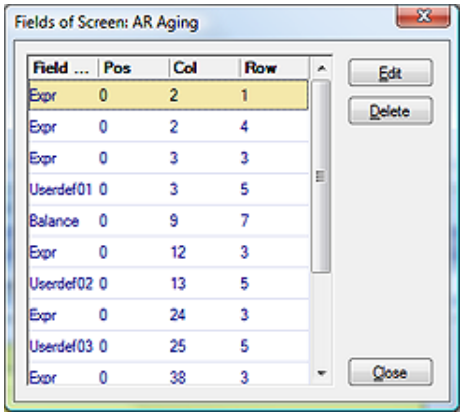

Figure 4-4

This brings up the **Fields of Screen:** dialog form similar to that shown here in Figure 4-4. About the only use that I found for this dialog form is to power delete fields from a screen. That is what I would recommend doing in this case. No organization will have any use for the screens as they come from the GoldMine default installation, although they are good examples of possibilities within GoldMine Premium. I do not recommend trying to layout out screens using this dialog form, as I prefer to visually see the layout as I am designing it. Highlighting the field to be removed, and clicking on the **Delete** button will do the trick. The user should hover over the **Y** key on the keyboard, and get into a rhythm of clicking on the **Delete** button, and then the **Y** key to remove the field from the screen. I always remove all of the fields, and I then begin with a new empty slate (screen). When I am through deleting all of the fields, I click on the **Close** button to return to the **Custom Screens Setup** dialog form.

Note

*Unlike past releases of GoldMine, in the **Tab Name:** field you are now permitted to utilize all 10 character positions, and all of those utilized will appear on the tab.*

*Additionally, whereas previously one had to utilize the underscore (_) to concatenate words for the GoldMine tabs, todays GoldMine will permit the use of spaces between words. Whereas, in the past we would have entered **AR_Aging**, today we can utilize **AR Aging** for a much more user friendly display.*

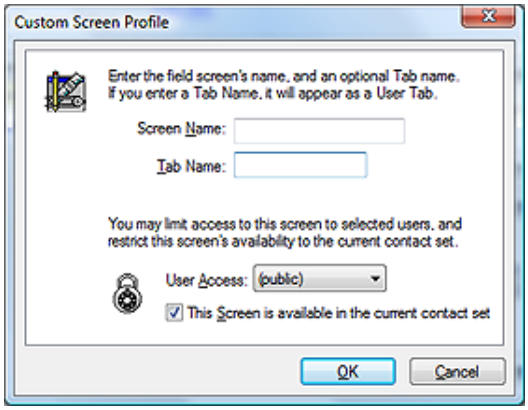

Figure 4-5

Next we should look at the dialog form presented when one clicks upon the **Properties** button as shown here in Figure 4-5. You may assign whatever name you desire in the **Screen Name:** field. This is the name that will appear on the local menu when the user right-clicks under the **Fields** tab. The next field, the **Tab Name:** field, on this dialog form is where one would assign this screen to a tab. To assign a screen to a tab, simply give it a name in this field.

Remember those **User Groups** that I discussed with you back in Chapter 2, well the **User Access:** field is one spot where you could employ a **User Group**. You may limit access to a screen to an individual GoldMine UserID or, to a group of GoldMine UserIDs. One must remember that this setting will not limit access to the screen, if the logged in user has Master Rights within GoldMine. Once access is limited, this screen will not even appear on the local menu as a possible selection choice unless the logged in GoldMine UserID is a member of the specified **User Group** that was given access rights. Should the screen have been defined as having a tab, that tab, as well, will only be available to those having access rights.

☑ **This Screen is available in the current contact set** is selected by default. One could easily wonder why this would even be here as an option. This screen information is maintained in the **Fields5** table which is common to all contact databases. This screen will appear in any newly created contact database. Though one may desire this screen in one particular contact database, they may not require it in another contact database. The GoldMine Administrator is **required** to edit this option for each screen in each contact database. Obviously, if you would want this screen in each contact database on your GoldMine system, then you would not need to edit this option. Editing is only required if you wanted to remove a particular screen from a particular contact database.

Once back at the **Custom Screens Setup** dialog form, you may click on the **New** button to add as many blank screens as you may have need for in your organization. It is usually considered better to have smaller screens of clustered information than one screen containing as many as 250 user defined fields.

The **Clone** button will exactly clone one screen into another, even containing the same name. This is best employed when trying to order screens for presentation to the user (see tip sidebar on previous page).

Now that you have your canvas (the screen) as well as your paints (the fields), you could begin to paint your picture. As I stated previously, I prefer to layout my screens graphically, although I must

warn you that this becomes a much more daunting task in GoldMine Premium. Large screens that used to take me 2 hours, can now take 3 and 4 hours to develop because of the idiosyncrasies within GoldMine Premium. Having said that, to do this, you would click on the **Fields** tab to bring them into focus. You would then right-click in the screen area under the tab, and select the screen that you would like to work on. Should you have assigned a tab name to a particular screen, you could go directly to that tab, and do your screen design in the area under that tab. In either case, once you are on the appropriate screen, you would right-click, and then select **Screen Design...** from the Local Menu. This will place your screen in design mode, showing row lines to aide you in field positioning. A toolbar will display itself as well. The toolbar will not appear in your design area as I show it below in Figure 4-6. Instead, it will position itself off of the screen while remaining easily accessible (maybe, I just had a hard time finding my toolbar). The toolbar icons, from left to right, permit you to place a **New** field, to **Save as** a Primary Field view, to **Load** a Primary Field view, to **Rebuild** to include new fields or to **Exit** the designer.

Note

*GoldMine Premium has the option to Customize the Designer toolbar, **NOT**. Even though the tools are there, the functionality is not present.*

Figure 4-6

The first type of field that I am going to insert is an **Expression** field. This type of field is used to display information only. The user will not be permitted to enter anything into this field as it is not a container (data input) field. We are not limited to the 15 characters that are allowed for labels of container fields either. In some cases, where I wanted an entire question to appear on my clients screen, I would add the question as an expression field, and then after it, add the container field without any label. You'll learn these various techniques as you get a few screen designs under your belt.

Note

*Even though the fields in the top portion of the GoldMine screen were initially laid out by FrontRange, you may change the attributes of the fields if you have Master Rights. Hold the **Ctrl** key down, and double-click on the field for which you wish to change the **Field Properties**. The result will be the dialog form, as shown in Figure 4-7, for the field you have selected. If you click on the **More options...** button, you will bring up the dialog form shown in Figure 4-8, which used to be the **Field Properties** dialog form, but today is known as the **Advanced Options** dialod form. From here you may change the label, the access rights, or any of the properties for these designed fields. One of the most common changes in this area is the **Field Access** properties.*

Remember that this is not changing the actual field properties in any way, and that this action is only changing the properties for the display of the field information.

I'll show you a screen shot for the expression field **Field Properties** selection, however, I'll talk about this tabbed Field Properties screen in more detail when I add the **ucSSN** field in a moment. For this, an expression field, look here at Figure 4-7. This is a change from past versions of GoldMine in that we must first give this expression field a **Label used for 'GM Dealer' record type:**, and, albeit disabled, the **Name in Database:** is displayed. Although, this is not signigicant for what we are currently discussing, it will play an important role when later we discuss **Record Typing**. For the time being, let's just click on the **More options...** button to bring us to, what is now titled, the **Advanced options** dialog form, refer to Figure 4-8 on the next page. In previous books this dialog form would have had the title: **Field Properties**.

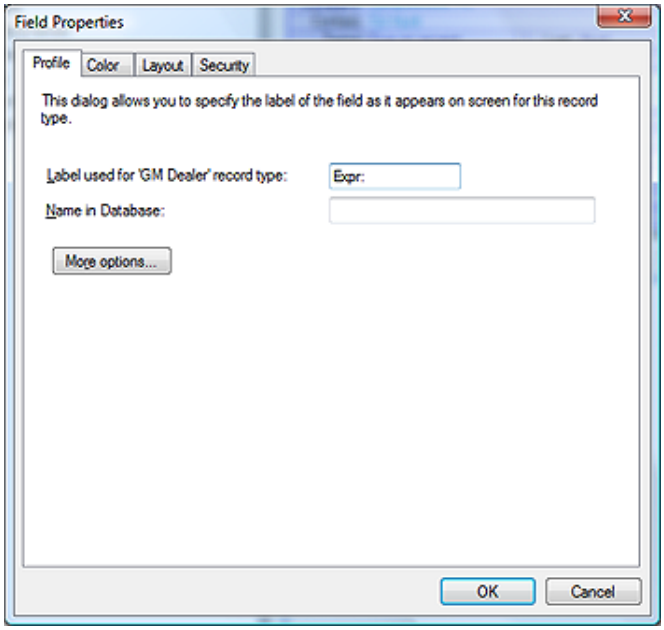

Figure 4-7

For this writing, I called this screen **Personal**, and it will contain personal information, in this particular case the **Social Security Number**. As one could have multiple screens represented under the

Note

It is a coding rule, when you are concatenating things together, that they must be all of the same type. You cannot concatenate a character type to a numeric type or a date type. Each type must first be converted to a character string before they can be concatenated together.

Note

*Any legitimate characters from your character set may be used in your expression as **chr(171)**. Here is a chart of my character set.*

--=== Extended Character Set ===--

33: !	47: /	168: ¨	183: ·
34: "	123: {	169: ©	184: ¸
35: #	124: |	170: ª	185: ¹
36: $	125: }	171: «	186: º
37: %	126: ~	172: ¬	187: »
38: &	127: ▯	173: ·	188: ¼
39: '	145: '	174: ®	189: ½
40: (	161: ¡	175: ¯	190: ¾
41:)	162: ¢	176: °	191: ¿
42: *	163: £	177: ±	222: Þ
43: +	164: ¤	178: ²	223: ß
44: ,	165: ¥	179: ³	
45: -	166: ¦	181: µ	
46: .	167: §	182: ¶	

Note

*Even though I have disabled Record Typing within my GoldMine the ⦿ **Global Label, used for 'GM Dealer' record type across all contact sets:** still picks up my Key1 field label **GM Dealer**. Your terminology of this statement may vary depending on your current GoldMine Record Typing setting.*

Note

*Even though you can spin the **Data Size:** field value up to **100**, when saved, GoldMine will re-adjust this value to **69**.*

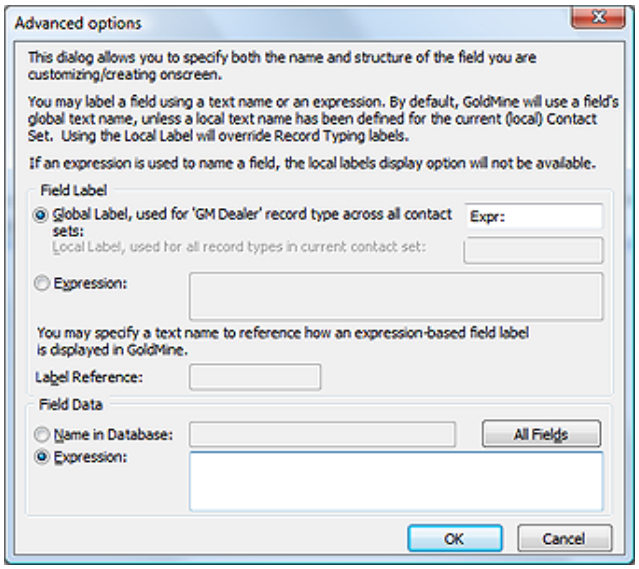

Figure 4-8

one tab **Fields**, one would want to have some sort of identification on each of the screens for the active screen if you are not using individual tabs. I will use an expression field to title this screen. I click on the toolbar **New** button, and I select **-- dBASE Expression --** from the list of fields (**Hint:** Just type a hypen (**-**) in **Field:** value of the **Place Field** dialog form). Clicking on the OK button from the **Place Field** dialog form will place the field on the screen. Double-clicking on the field on the screen, or keying the Enter key from the keyboard to bring up the Field Properties dialog shown in Figure 4-7 on the previous page.

To arrive at the **Advanced options** dialog form as of GoldMine Premium 8.5.1.12, one must click on the **More options...** button, Figure 4-7. Notice that you are positioned on the radio button ⦿ **Global Label, used for 'GM Dealer' record type across all contact sets:**, and the field contains **Expr:**. There has been a slight change here for GoldMine Premium, as GoldMine will no longer allow label names to remain blank. You must have something in the **Field Label** frame before you can leave the **Advanced options** dialog form. In order to leave the label blank, you need to trick GoldMine. I have found that if you place a colon (**:**) in this field, that this will suffice, or you can just leave the default **Expr:** value.

Next, you'll need to add your expression. This is the expression that I put into the **Field Data** frame ⦿ **Expression:** field:

```
[--===| Personal Screen for ]+trim(Contact)+iif(empty(Company), [],[ of ]+trim(Company))+ [ |===-- ]
```

Though the expression field appears to be large, and appears as though it will wrap text, it does not (go figure as my spouse would say). This expression may or may not be appropriate, but it does give you an example of what can be accomplished with an expression field. The nice part is that expressions are always evaluated in real time, therefore, as you change contact records, the expression field will change accordingly. I will give you a better example of how appropriately an expression field may be applied, when I discuss an expression field against the date of birth field later in this chapter. That expression field will be to display the age of the contact in real time.

Being an old dBase programmer, I prefer to use the square brackets (**[ ]**) when possible instead of the quotation (**" "**) marks to delimit a character string. In pseudo code (programmers use pseudo code a lot), the expression represented above concatenates a character string with a character field that has had the spaces trimmed off of it, with a function that evaluates the company field. If the company field is empty, the function adds nothing to the expression while if the company field contains information, the function concatenates a character string and the company name. Wow, that was long winded.

Let's look at the individual pieces.

[--===| Personal Screen for]	character string
trim(Contact)	trim function on character field
iif(empty(Company), [],[of]+trim(Company))	immediate if function returning character string
[|===--]	character string

The plus (**+**) sign is used to concatenate character strings together. To be correct, and to be displayed in GoldMine, the expression must result in a valid character string. For example, if we had added the age() function to the end of this expression, it would have resulted in expr error being displayed on this screen. The age() function returns a numeric value from a date field. This would have violated the rule that "The result of any expression must return a character string".

Next, we should move on to the **Layout** tab, Figure 4-9 on the next page, and we will make some adjustments here. First, in the **Field Size & Position** frame, specifically in the **Field Label Size:** field, as I don't have a label for this expression field, I will drop this value from 10 to **0**. I will then make the **Data Size:** value **69** (the maximum allowed in this field). For this exercise, I will set the **Colon (:) Row (y):** value to **15**, and I will then set the **Column (x):** value to **2**. I can now click on the **OK**

button to accept this expression, and its layout. Based on the contact record that is currently active in my GoldMine Premium, the expression immediately displays:

--===| Personal Screen for DJ Hunt of Computerese Inc. |===--

You should have noticed that the expression is evaluated immediately upon saving the expression even though you remain in the design mode. This is called instant gratification as well as letting you know that your expression is correctly syntaxed.

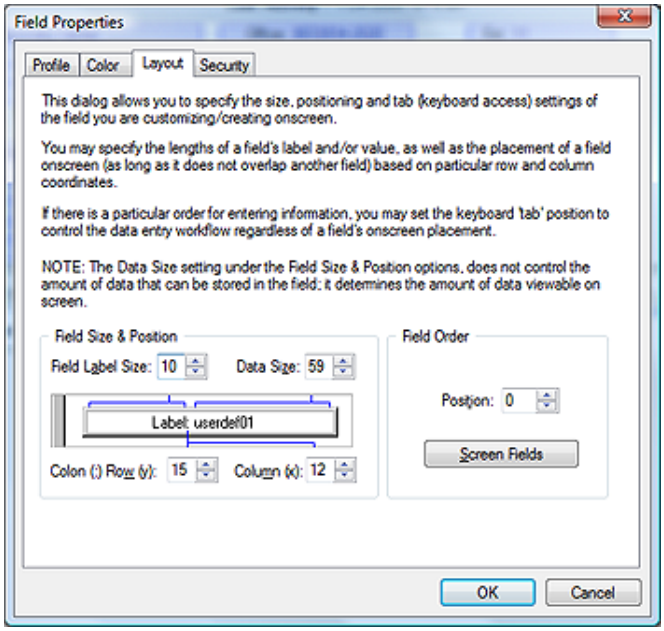

Figure 4-9

Now let's look into adding a field to the screen, and I'll make use of some of the new features that allow for field level record typing. I'm going to add the field to the **Personal Screen**, but I only want the field displayed when the record type is **Buyer**, **Seller**, or **Agent**. If the record type is **Property**, then I do not want the field to be displayed. I am using the **Contact1.Key1** field for the record typing driver. How about changing the color of the label, and the color of the data when the fields are displayed to, let's say, blue for the label and magenta for **Buyer**, red for **Seller**, and green for **Agent**. I have already added a new custom field named **ucSSN**, **C**, **11** and, while I was at it, I added another field **udDOB**, **D**, **8** (we'll use this one later in this chapter).

Add the field to the screen as I did with the expression field before, only this time select the field **ucSSN** (as it appears on our list, yours may show Social Security Nu (Ucssn)), and then double-click on the field once it has been displayed on the screen. Now let's change the **Label used for 'GM Dealer' record type:** to **SSN:**, and then click upon the **More options...** button. The resulting **Advanced options** dialog form, for this exercise, is shown here in Figure 4-10. Please note that the ◉ **Global Label, used for 'GM Dealer' record type across all contact sets:** should have been populated with the field description placed in the **Label used for 'GM Dealer' record type:**. What I show you in Figure 4-10 is what I have used for a **Global Label**. I have inserted the label **SSN:** as described earlier, which is what I would want displayed as the label when the field is displayed on a screen. Click on the **OK** button, to return to the **Field Properties** dialog form.

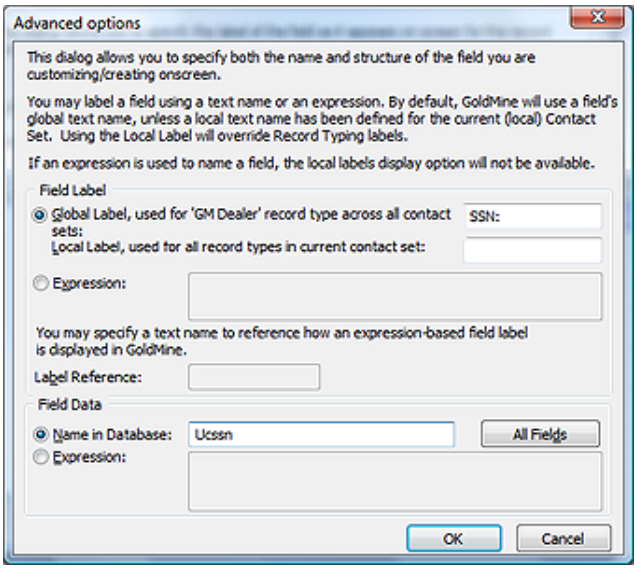

Figure 4-10

Next click on the **Color** tab, and you can see the results of this exercise in Figure 4-11 on the next page. Remember that I said that this field should show for all field level record types except **Property** (I don't know of any property that possesses a Social Security Number), and then, when the label was displayed, that it would be Blue (see sidebar **Tip** on the next page for determining the numbers to use to produce the desired colors).

In order to achieve my goal of label and data not being displayed when my record type is **Property**, and to be displayed with a **Blue** label otherwise, as well as **Magenta** for the data field on a record type of **Buyer**, **Red** for the data field on a record type of **Seller**, and **Green** for the data field on record

Note

Global Label *vrs* ***Local Label***

You might ask: "When should I use a ***Global Label*** *as opposed to a* ***Local Label****?".*

When you create a new Contact Set based on the structure of an existing Contact Set, the Global Label is carried across to the new Contact Set, however, the Local Label is not.

In theory, the Local Label is per Contact Set, while the Global Label is, well Global, and applied across all new contact sets.

Note

Social Security Numbers *usually take of the form of* ***###-##-####****. In the* ***Lookup.ini*** *chapter of this book, I explain to you how to create this presentation regardless of what your end users enter into this field.*

Another formatting option that I discuss for the Social Security Number in the Lookup.ini chapter is the ***GMTray*** *application.*

Tip

I can never remember the numbers assigned to the various colors. When I get on the ***Colors*** *tab, I immediately select the* ***Colors...*** *button. I then select the first of the colors that I want to use from the color chart, and select* ***OK****. I then write down the number that appears in the ○* ***Expression:*** *window, and I will use that number in my expression. I do this for all of the colors which I intend to be using.*

Note

*When setting colors, **-1** represents the default GoldMine color for that attribute, while **-2** hides the entity on the display. To hide a field entirely from the display, the **Label Color**, and the **Data Color** must both be set to **-2**. All other colors must be represented in the expression by a valid color number.*

Tip

When creating logical expressions, try to account for user unpredictability. For instance a user could enter into the Record Type field:

property
Property
PROPERTY
PrOpErTy

Anyway, you do want your expression to pass True for all of these cases, hence, I trim() off any trailing spaces, and convert the value to all upper() case for comparison.

trim(upper(Contact1->Key1)) == [PROPERTY]

*You may have noticed that I used the exactly equals operator (**==**) as well.*

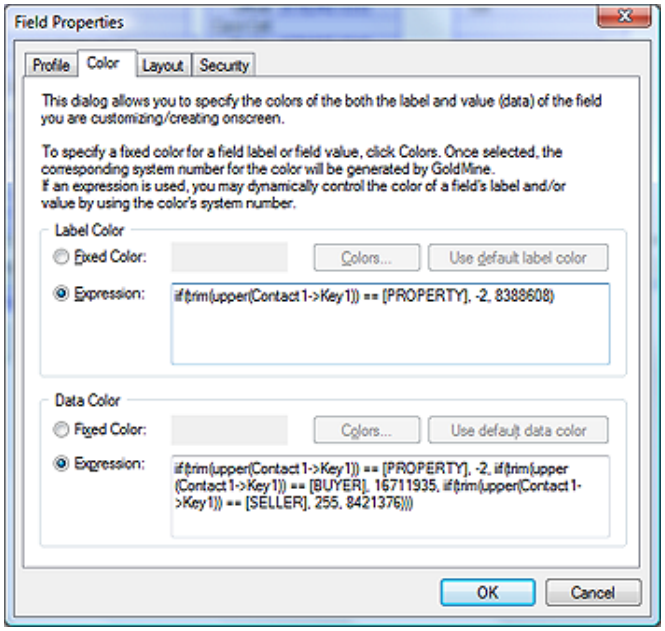

Figure 4-11

type of **Agent**, I must employ the O **Expression:** option for each. Each expression employs the **i**mmediate **if** function, iif(logical result, true, false). In fact, for the **Data Color** expression, I've embedded three levels of this function, however, let's examine the **Label Color** expression carefully first.

iif(trim(upper(Contact1->Key1)) == [PROPERTY], -2, 8388608)

My logical expression is:

trim(upper(Contact1->Key1)) == [PROPERTY]

Should that evaluate to True, I set the color to:

-2 (Hide)

Should that evaluate to False, I set the color to:

8388608 (Blue)

Are you ready for the big one? Here is my explanation of the Data Color expression:

```
iif(trim(upper(Contact1->Key1)) == [PROPERTY], -2,
iif(trim(upper(Contact1->Key1)) == [BUYER], 16711935,
iif(trim(upper(Contact1->Key1)) == [SELLER], 255, 8421376)))
```

Keep in mind that this is one continuous line of code, and is only separated here for display purposes.

My 1st logical expression is:

trim(upper(Contact1->Key1)) == [PROPERTY]

If that evaluates to True, I set the color to:
-2 (Hide)

If that evaluates to False, evaluate the 2nd logical expression:

trim(upper(Contact1->Key1)) == [BUYER]

If the 2nd logical expression evaluates to True, set the color to:

16711935 (Magenta)

If that evaluates to False, evaluate the 3rd logical expression:

trim(upper(Contact1->Key1)) == [SELLER]

If the 3rd logical expression evaluates to True, set the color to:

255 (Red)

If the 3rd logical expression evaluates to False, set the color to:

8421376 (Green)

I didn't need to test for **Agent** as, in our scenario, that was the only possible remaining record type. However, should you have more record types, then you would have evaluated for **Agent**, and on false, you might have set the color to the default color. I won't analyze it for you, but that expression could look like:

```
iif(trim(upper(Contact1->Key1)) == [PROPERTY], -2,
iif(trim(upper(Contact1->Key1)) == [BUYER], 16711935,
iif(trim(upper(Contact1->Key1)) == [SELLER], 255,
iif(trim(upper(Contact1->Key1)) == [AGENT], -1, 8421376)))
```

I haven't tested this out for you, so I can't state for certain, however, in the dBase language one is not able to embed deeper than 25 levels. In the code above, I have embedded 4 levels. You'll need to test this out for yourself if you are truly interested in seeing the depth to which you are allowed to embed iif() functions.

Let's move on to the **Layout** tab, as displayed here in Figure 4-12. Here we can see the **Field Size & Position** frame. Users tend to misunderstand this section. Any changes made here are not, in fact, making changes to the underlying data (field) structure, but are, instead, making changes to the portal that displays that information. In support of that statement, the ⦿ **Global Label, used for 'GM Dealer' record type across all contact sets:** can be as many as **15** characters long. To display that many characters on the screen one would need to increase the **Field Label Size:** from **10** to possibly as much as **15** (this is solely dependant on screen font). Changing this number to **20** would not gain one any more label space than the data field currently holds. The **Data Size:** might have been a better example. You may have a field that is 256 characters long, while the **Data Size:** should not exceed **69** (this assumes that you have a **Field Label Size:** size of **0**, see sidebar). If you are using the small fonts in GoldMine, this means that about **100** characters of your **256** character long field will be displayed. In the edit mode, you may access the rest by scrolling, however, what you are designing here is the portal to the data, and what it will display. Again, don't get the data field structure size confused with this portal display view size.

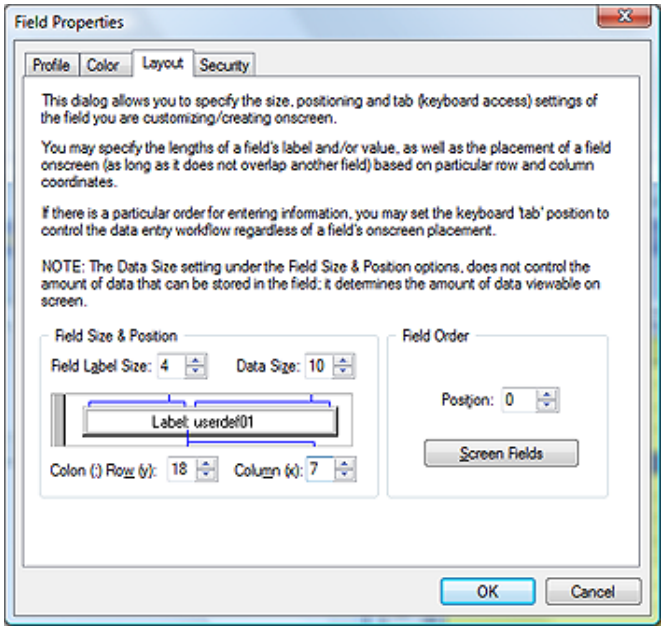

Figure 4-12

Without rehashing what I've already said a number of times, the **Field Label Size:** is where you indicate how much of the label is displayed on the screen. The **Data Size:** represents how much of the data field structure is displayed through the portal.

The bottom row of information defines where that portal is positioned on the screen. The positioning on the screen is defined based on where the **Colon (:)** would be positioned. To emphasize, the positioning is not based on the beginning of the text or the end of the text, but, instead, from where the colon between the label and the data is to be positioned on the display. Usually, on your screen, you would have a label and data, it would be displayed thusly, **Label: Data**. Though you don't always have a label (when you want to have a question followed by an answer and the question is more than 15 characters in length for example) the positioning always assumes the colon position proceeds the data, even if no label is attached.

The first **Row (y):** available under the **Fields** tab is row **15**, however, that is extremely close to the frame, and placement on this row would not be visually appealing. On the other hand, the **Column (x):** may be anything from **1** to whatever. That whatever can be a gotcha as well. You could, in theory, place the field on the screen where no one could get at it. Let's say that you made the **Column (x): 200**. You would be able to see the field move out of view into the nether world while you are in the design mode, and this is exactly what the users would see. Yes the field is there, but it is neither visible nor accessible. As a designer, you need to be cognizant of your every move. Keep that **Column (x):** position to the viewable screen position on a standard resolution monitor.

The next frame, on this tab, is the **Field Order** frame. In this frame the designer determines the tabbing order for this field as compared to the other fields in the view. The tab order should not even be available for an expression field, as the user could not possibly tab into or out of an expression field on the screen, nor edit the information contained therein. However, it is available, and I would certainly recommend that you do not even set the tab order for expression fields. If you are placing an

Note

Field Label Size: and Data Size: do not relate one-to-one with the number of characters. One may require a Field Label Size: of only 12 to display a label containing 15 characters.

Note

GoldMine will enforce its rule that the Field Label Size: plus the Data Size: may not exceed 70. Even though you can spin the number high, when saved, these will be reset to sum 70.

Note

Remember to allow for the F2 Lookup arrow that follows the data portal. I usually allow 4 places for the arrow. If I had a two character field, and I wanted to display both characters, I might increase my Data Size: to 6.

Note

If you require a Label that is larger than the 15 characters allowed by GoldMine then you are advised to utilize an Expression field for the label as I had discussed earlier in this chapter.

Tip

Personally, I wait until the screen is totally laid out, at which time I return to set the Tab Order. I feel that this saves a lot of extra steps.

editable field on the screen, however, then it is wise to set the tab order, **Position:**, for those fields. To us, the term **Position:**, is confusing to the designer. Do not get confused, setting this number is setting the **tabbing order** number, and tabbing is done in sequence. The spinner control to the right of the field will increment the number from **0** to **100** in increments of **1**. As you could have up to **250** fields per screen, if you wanted to change the tab order for all 250 fields, you would be required to type the tab order by hand beyond the 100 spinner number limitation.

Tab Order - When one has placed a field on the user defined screen in edit mode, and then depresses the **Tab** key on their keyboard, the cursor will advance to the next field in the sequential tab order.

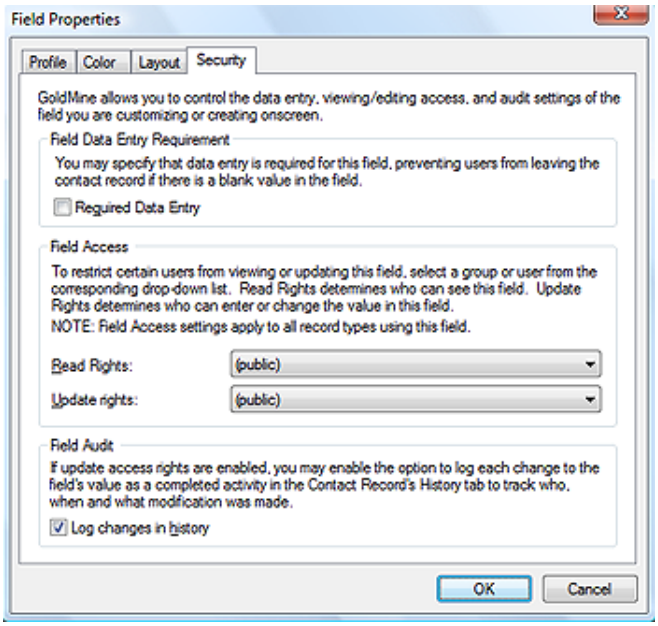

Figure 4-13

We can now continue on to the next tab, that being the **Security** tab, Figure 4-13. The first frame, **Field Data Entry Requirement**, contains but one option, ☐ **Required Data Entry**. As the dialog states: *"You may specify that data entry is required for this field, preventing users from leaving the contact record if there is a blank value in the field."* I believe that defines this satisfactorily.

The first selection, in the **Field Access** frame, is the **Read Rights:** field, and then the **Update rights:** field. This is the same type of security that we had access to when we were setting up the screens, however, this access is applied at the field level as opposed to the screen level. If you wanted a certain user or group of users to be able to see this field, then you would assign that user or group of users to the **Read Rights:** field. The same applies to the **Update rights:** field. A typical case may be that you want some users to be able to see a field, but you might not want those same users to have the ability to update the information in that field. The default for each field is **(public)**, which, of course, means everyone has the ability to see, and to update the data value in the field. Remember, in either of these cases, you are assigning the user(s) that will have **Read Rights:** and **Update rights:**. Field Access rights are a very good reason for defining your user groups (discussed earlier in Chapter 2) appropriately, and remember that users may belong to more than one user group. Take your time when designing your user groups. You can always modify them later, but make them as useful as possible when you begin your design. As this is the Social Security Number field, you may want only certain individuals to enter this information into the field (**Update rights:**).

Note

*You may remember that I discussed the UserID **Access** property **Required field override** option back in Chapter 2. Combined with this option, the option discussed earlier may now supply you with a better understanding of Security policies within GoldMine.*

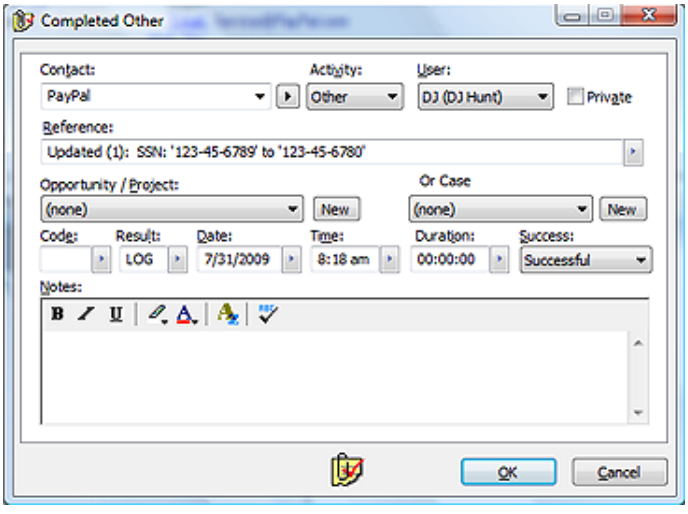

Figure 4-14

The next frame, on the **Security** tab, is the Field Audit frame which, again, contains a single option. This option is the ☐ **Log changes in history** option which is not checked in its default state, however, for the SSN I have chosen to select it. It would have been good practice on the part of FrontRange to have disabled this option when creating an expression field. By their very nature, an expression field cannot be changed. However, as this is a field that I am placing on my screen, I may want to check this

Note

*In past versions of GoldMine, selecting the ☐ **Log changes in history** option, would not have created a History record had the field been blank prior to the field change.*

As of GoldMine Premium 8.5.1.12, FrontRange has rectified this. Whether the field is blank or contains a value, when modified, a GoldMine History activityis created.

option. The Social Security Number is a unique number, and shouldn't change ever, once entered. I would want to know what, when and who changed this field if it were ever to change. Any change in the field being monitored would result in the creation of a History Activity under that contact record with all of the pertinent information.

Figure 4-14, previous page, is the result of a change made to the **SSN:** field that had been set to ☐ **Log changes in history**. GoldMine shows you the userID that changed the field, as well as the date and time that the field was changed. Notice also, that GoldMine maintains the old value as well as the new value of the field being monitored for change in the History **Reference:** field.

```
Updated (1):  SSN: '123-45-6789' to '123-45-6780'
```

Selecting this option for more than a couple of critical fields could result in a fast growing **ContHist** table, and therefore, I do not recommend that you abuse this field monitoring convenience especially if your remote users are working against the SQL Server Express 2005/2008 backend with their database size limitations. Select this option for your most mission critical fields, and only for as long as monitoring is deemed necessary. With the proper security settings, monitoring a field for changes should not even be required unless you suspect that there is a problem within your company.

Earlier, I mentioned that I would add a field on the **Personal** screen, **udDOB**. If you are doing the same, I would like you to set the Color attributes exactly as I had done for **ucSSN**. You see, I would not want this field to be displayed if the **Record Type** were that of **Property**. Now if one assumes that this is the persons **Date of Birth**, one may want to display the individuals age on this screen so that a user reviewing the information will not have to pause to calculate the individuals age while in the middle of a conversation with that individual. To do this, I would ask that you add an expression field. This is the expression field that I employed:

```
iif(dtos(Contact2.uDOB) > [], [Age: ]+ltrim(str(age(Contact2.uDOB))), [Age: Date of Birth Empty])
```

Remember to put a colon (**:**) into the **Label used for 'default' record type:** field. Remember to set the **Color** attributes identical to those of **ucSSN**. I would not want this age field to display for a **Property** record type, though I might create an alternative to display, showing the age of the property.

As a quick source:

Label Color

```
iif(trim(upper(Contact1.Key1)) == [PROPERTY], -2, 8388608)
```

Data Color

```
if(trim(upper(Contact1.Key1)) == [PROPERTY], -2,
iif(trim(upper(Contact1.Key1)) == [BUYER], 16711935,
iif(trim(upper(Contact1.Key1)) == [SELLER], 255, 8421376)))
```

Lastly, remember to set the **Field Label Size:** from **10** to **0** as we are generating our own label, **Age:**.

Now, whenever the user is on a record, of **Record Type: Buyer**, **Seller**, or **Agent**, and there is a value of **11/23/1948** in the **udDOB** field, the age will be displayed as:

```
Age: 60 ( at least it was on July 31st, 2009 )
```

With no value in the **udDOB** field, the age will be displayed as:

```
Age: Date of Birth Empty
```

Here a little tidbit that I discovered in the forum. To change the color of a single word based on the 1st character of the word, you could try this expression:

```
color(word('Black Blue Red Green Yellow Purple Orange Gray', at(left(Contact1.Key1,1), 'ABCDE-
FGN')))
```

Notice that I am applying this based on the 1st character in the Key1 field. I would also like to mention that, although it does work, it does not work as expected in GoldMine Premium. The entire expression value is converted to the color based upon the 1st letter in the Key1 field.

Record Typing

What exactly is **Record Typing**?

Well, I have been discussing **Record Typing** of sorts in this chapter already without specifically pointing a finger to it. I have been discussing **Field Level Record Typing**, and this was my preferred method of **Record Typing** within GoldMine. There is, however, **Record Level Record Typing** inherent within GoldMine Premium which has improved tremendously since its inception in the GoldMine

product. In its basic terms, the GoldMine Administrator would designate one GoldMine field as the **Record Type** field. I'll work with record typing as designed for use in an automobile dealership as my subject matter for this explanation (this is the sample configuration GoldMine Premium). Therefore, if I were to use, let's say, **Contact1.Key1** as the **Record Type:** field, I would then create an accompanying **F2 Lookup List**, refer to Chapter 6, that contained **Buyer**, **Service**, **Agent**, and **Vehicle**. Others could be added to this list, however, this should suffice for this section of the chapter. I would then want to lock down that list to not ☐ **Allow blank input**, and to ☑ **Force valid input**, while it might be nice to give a **Field Name:** for this F2 Lookup List of, oh let's say, **Record Type**. I might also want to check the option to ☑ **Auto Fill**. This field would then be employed to designate what type of information a particular record is going to contain. For example, if the **Record Type:** field contained **Buyer** on a particular record, then one would know that all of the information contained on this record, as well as the screen design, would belong to the **Buyer** record type. Whereas, if this field were to contain **Vehicle** on a particular record, then one would know that the screen design and fields would represent the vehicle on the lot to be sold or delivered. By the way, in this scenerio, the Relationship Tree is your best friend. You could relate an Agent, to a Buyer, to a Vehicle for easier visualization. How sweet (hmmm - efficient) is that?

All other GoldMine **Contact1** and **Contact2** table field **Labels**, **Color** and **Display** may be based upon this one field. I have discussed **Field Level Record Typing** in more detail in the **Screen Design** section of this chapter, however, it is important that one understand the concept now. As well, it is important to understand that the combination of **Field Level & Record Level Record Typing** is highly recommended for the best possible presentation of **Record Typing**. Additionally, which view displays in default under **GM+Views** may also be based upon the type field. Therefore, based on this one field, the end user could see pictures of the vehicles, if the record were designated a **Vehicle** type, or the end user could see the buyers interest and history if this were a record type of **Buyer**.

Let's take a look at the default **Contact1.Company** field. This field is a **Character** based field having a field length of **40** characters (refer to Chapter 9). No matter what one does with the display of this field, the underlying characteristics of this field will never, ever be changed. That is an important concept to accept in your understanding of record typing. When looking at the GoldMine display, and when having selected either **Buyer**, **Service**, **Agent** or **Vehicle** in the **Contact1.Key1** field, it might be appropriate to have this field label display **Buyer:**, however, if the **Contact1.Key1** field were to contain **Vehicle**, then it might be more appropriate to have this label now display **Vehicle:**. Mind you, the underlying field will not have changed, it is still **Contact1.Company**, **Character**, **40**, however, it would be presented to the end users as **Buyer:** or **Vehicle:** depending upon the value in the **Contact1.Key1** field. As well, I could easily change the color of the **Label** or **Data** in the display based on the value in the **Record Type:** field.

So, what is **Record Typing**?

Well, Record Typing is nothing more than the ability to use one table for different types of data input. FrontRange has been using this concept for years without the end user being fully aware of this. Yes, the **ContSupp** table employs a field called **RecType** that determines what type of GoldMine information is contained in any given **ContSupp** record. You'll read about this later in Chapter 9, however, let me show you this ahead of time:

■ The following are possible values for the **ContSupp.RecType** field:

A	**Record Alerts**	**L**	**Linked document**
C	**Additional contact record**	**O**	**Organizational chart**
E	**Automated process attached event**	**P**	**Profile record/extended profile record**
H	**Extended profile header**	**R**	**Referral record**

So one can easily see that GoldMine is using the **ContSupp** table to store **Record Alerts**, **Addition Contact Records**, **AP's**, etc., and all without change to the underlying table structure. GoldMine takes this hard coded record typing to the user level, and let's you apply this same concept against the **Contact1**/**Contact2** tables as well as **GM+Views**. You may designate one or more fields to define your Record Type, and based on the Record Type the, let's say, Company field could contain a company name, or a part number, or anything that you design dictates. Very neat customizability indeed.

Up until this book, that which was previously stated was the extent of the Record Typing section of this chapter, and, although I still feel that it has issues, I will expand upon Record Typing for the GoldMine Premium edition of this book. That's right, here and now.

One must be a user possessing Master Rights in order to select

Tools
Configure ►
Record Types

from the GoldMine menu which, by doing so, will return a dialog form similar to that which I am showing below in Figure 4-15. I have expanded all of the trees so that you will have a clearer picture of the **Record Type Administration Center** as it has been distributed with the GoldMine Premium Edition.

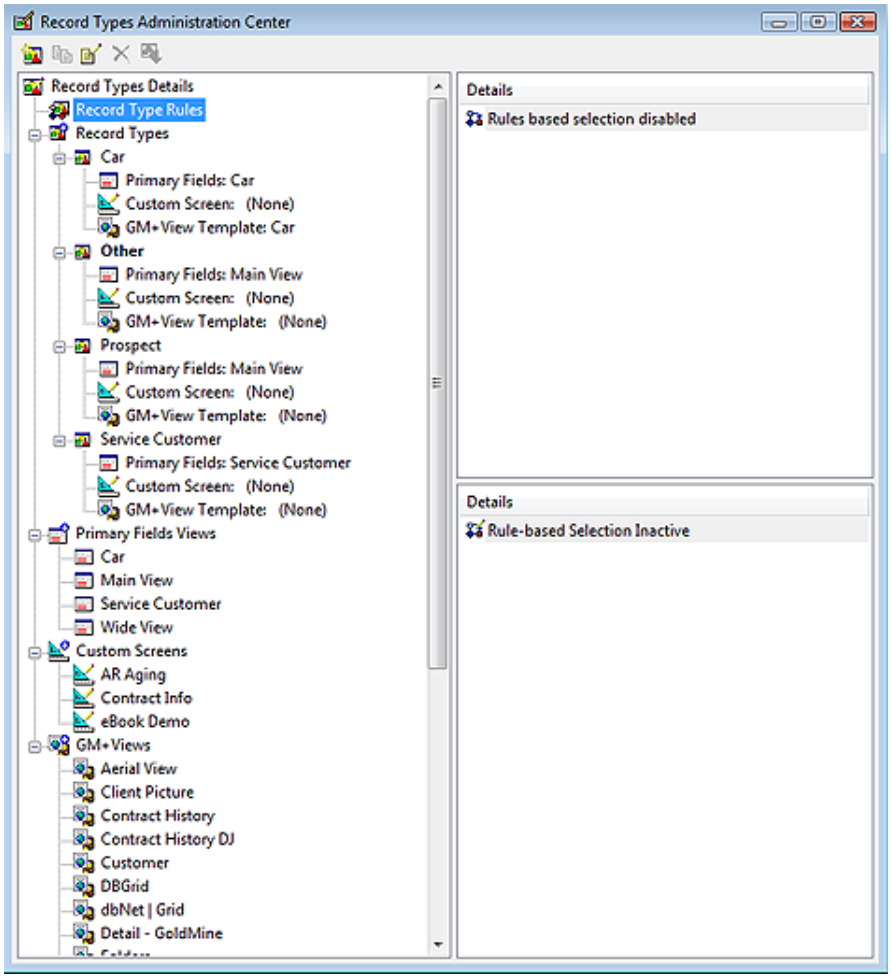

Figure 4-15

To achieve the most that you can out of the Record Type Administration Center capabilities, you want to work backwards, in effect, climbing your way up the tree. In fact, prior to even entering the Record Type Administration Center, you would want to create any and all of your potential GM+Views, and Custom Screens. The Custom Screens are only necessary if each screen is unique. If, however, you plan on utilizing one screen for different Record Types (recycling fields) then you would use the Clone capability of the Record Type Administration Center. Wow! I think that I confused myself there. Simply put, unique screens are developed with GoldMines Screens Setup... utility while re-used screens will be designed via cloning in the Record Type Administration Center.

Under the **Custom Screens**, shown above in Figure 4-15, you will see that I have already created 3 distinct custom screens: **AR Aging**; **Contract Info**; **eBook Demo**. Let's begin by cloning the **Contract Info** screen. This is a screen that I had created earlier for our companies actual GoldMine usage. Right-click on the Contract Info screen, and select **Clone** from the local menu. Your resulting dialog form should be an old friend, the **Custom Screen Profile** dialog form. Notice in Figure 4-16, that I have added a screen name of **Record Typing Sample**, and a tab name of **Sample** for this exercise. Clicking on the **OK** button will then add the **Record Typing Sample** screen to the **Custom Screen** tree, Figure

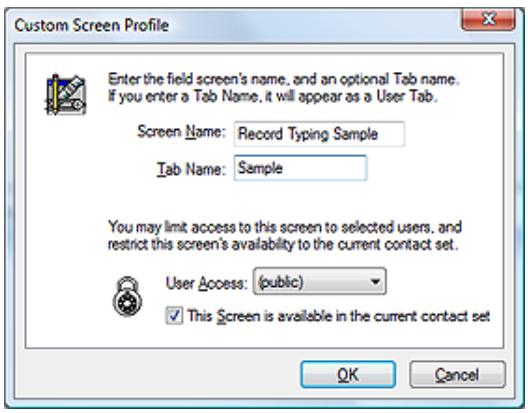

Figure 4-16

Note

*You cannot add fields to a screen via the **Record Types Administration Center**, so plan ahead. Begin with a User Defined screen that has many fields on it, and then **Clone** it for your new screen with reusable fields. Field reuse is the key for Record Type screens.*

WARNING

Issue:

Using Local Labels in Record Typing

***Response:** (Grant Ellsworth/2008)*

I found evidence that Local Labels are not associated with the specific Record Types.

Here's how/why:

The Record Types, and the associated field labels or Global labels, are stored in the Fields5 table (refer to The Tables chapter); Local Labels are stored in the database's ContUDef table, making the Local Label global to all record types. Hence, I use only Global Labels for the Record Type labels. This consistently works.

Hence, Local Labels and Record Typing are a non-intersect set, an oxymoron, if you will.

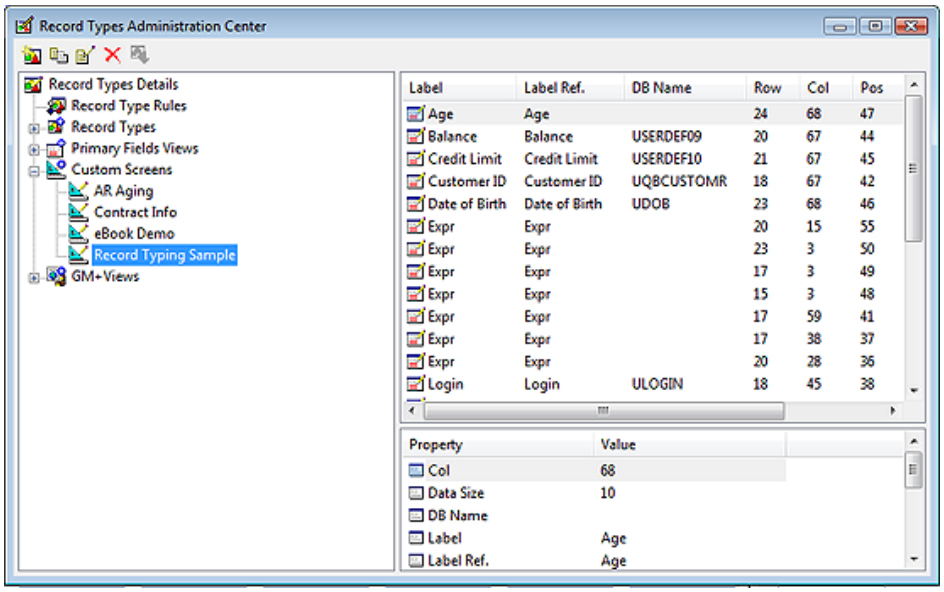

Figure 4-17

4-17. If the **Record Typing Sample** is not already highlighted, please highlight it now. If you look in the upper right hand side of the dialog form, you should see all of the fields/expressions that we had previously added to the **Contract Info** screen repeated exactly on this screen. Below that, if a field is highlighted, you will see the **Properties** associated with the highlighted field.

If you right-click on, let's say the **Tilde (~) Usage** field (not shown in Figure 4-17) or any field for that matter, and then you select **Edit...** from the local menu. Next, if you select the **More Options...**, you would bring into focus, our old friend, the **Advanced options** dialog form. To understand this, you must position yourself properly within your mindset. You are editing some of the field properties for this field only as it relates to the **Record Typing Sample** screen. You will, for instance, be able to change **Field Label** as I have done here in Figure 4-18. You will also be able to change the **Field Data**, so although you cannot add a new field to this screen, you will have the ability to change the data field to another predefined field or expression. You will have full access to all properties on the **Color** tab via the **Field Properties** dialog form. You will not have the ability to change any of the properties on the **Layout** tab except for the **Field Order**. You will have full access to all properties on the **Security** tab. Let me reiterate, however, that any changes that you make via the **Field Properties** or the **Advanced options** dialog form here, are only relevant to that field when the end user is on the **Record Typing Sample** screen. You have to always remember and think of this as field recycling.

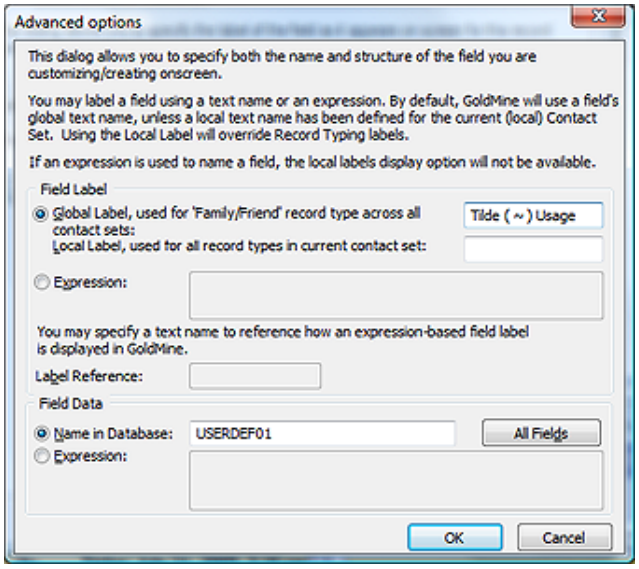

Figure 4-18

Let's climb up that tree a bit further now to the **Primary Fields Views** branch. GoldMine Premium comes standard with four primary views: **Car**, **Main View**, **Service Customer** & **Wide View**. You may remove any of these, if they are not necessary in your schema, by right-clicking on the view, and choosing **Delete** from the local menu. You may want to keep these here as examples. The choice is strictly yours. I just had to figure a way of discussing the ability to delete views that exists in the **Record Types Administration Center**, and to that end I have succeeded.

Now let's say that you had your own record type, and that you wanted to reutilize a screen to define your own **Primary View** for that particular record type. Again, right-click on the view that you wish to use as your base, and select **Clone** from the local menu. This is nothing different than that which

we just accomplished for the new **Custom Screens**.

Shimmy on up that tree another branch to the **Record Types** branch. Let's highlight that branch by selecting it, and then right-click and select **New Record Type** from the local menu. The resulting dialog form is shown here in Figure 4-19. It is here that we can bundle our **Record Types** into a package that could include one leaf from each of the underlying branches.

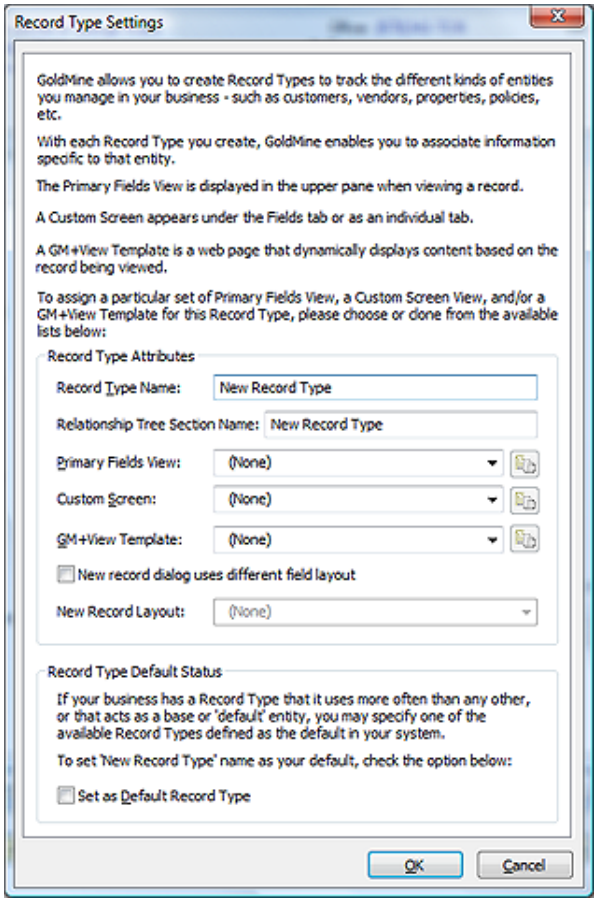

Figure 4-19

In the first frame, **Record Type Attributes**, we have to make our choice of a view for each category, however, first you would want to choose a **Record Type Name:** for your bundle. As an example only, I will type **Sample Record Type** into this field. You feel free to do as your creative mind instructs you to do if you are following along with this example that is. Now we must instruct GoldMine as to what screens to utilize as the default screens for this record type. The first screen necessary would be the **Primary Fields View:** screen, then on to the **Custom Screen:**, and, finally, the **GM+View Template:** screen. In all cases, you will only be able to select a single screen from your already defined screens contained in each branch.

Note

*Notice that I said the **default screens** for the Record Type. Obviously the user could select from any of the available **Custom Screens** or **GM+Views**, hence, I am talking about the default state when a user sets focus to a record of that particular record type.*

WARNING

*You are strongly advised to **not click** on the pages icon at the end of the **GM+View Template:** field if you are utilizing GoldMine Premium 8.5.1.12. In no way does it function as expected, and it will only add more GM+Views with the same name yet with an incremented number as:*

WebSite
WebSite (1)
WebSite (2)

Let's use the **Primary Fields View:** for this example. Once you have select a view to utilize for this record type, you may further modify its properties by clicking on the pages icon to the right of the view name which, in turn, will pop this **Main View Profile** dialog form shown here in Figure 4-20. The dialog form title will vary depending on the view that you are setting such as the **Custom Screen Profile**. It is from this dialog form that you may set a **Name:** for the main view or custom view, depending upon which dialog form you are in, and you may also limit the access to this view to a single UserID or a GoldMine User Group via the **User Access:** drop list.

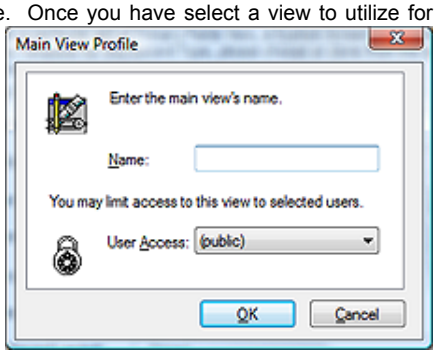

Figure 4-20

Returning to Figure 4-19, you will notice one more option in the **Record Type Attributes** frame, that being ☐ **New record dialog uses different field layout**. Although grammatically lacking, what this option is saying, when selected, is that one would like to use a different **Primary View** when they are creating a new record for this record type. If you do select this option, then you will be required to select the alternative **Primary View** to be utilized when creating a new record of this type.

There is one more frame that we have to consider in this dialog form that being the **Record Type Default Status** frame. You may only select this option for one record type in your GoldMine. As it clearly states on the dialog form: *"If your business has a Record Type that it uses more often than any other, or that acts as a base or 'default' entity, you may specify one of the available Record Types defined as the default in your system."* If this is to be that **Record Type**, then you would simply check the ☐ **Set as Default Record Type** option. Now, by clicking on that **OK** button, you will have defined your own **Record Type**. You may now continue on and design as many record types as you will require for your organization.

But wait, we're not done yet. GoldMine doesn't, as yet, know when to apply these **Record Types**. Climb up that tree just one more branch now, will ya? Right-click on the **Record Type Rules** branch, and select **Edit...** from the local menu to bring up the dialog form shown, next page, in Figure 4-21.

Tip

It is a good practice to disable your record typing rules while designing the various record types.

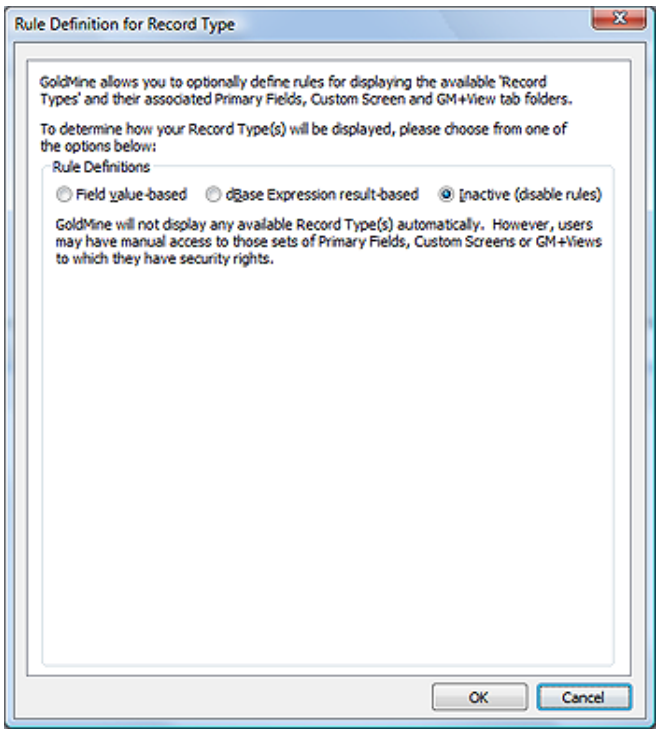

Figure 4-21

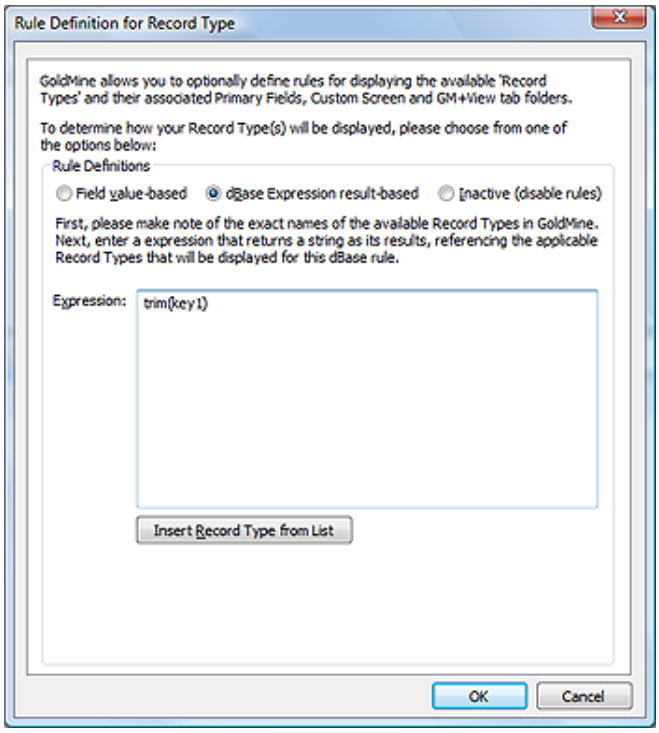

Figure 4-22

It is here that GoldMine allows you to define the Rules by which the different record types are displayed with one of the possibilities being to make the rule O **Inactive (disable rules)**, however, doing so would leave me with nothing further to discuss. Oh, and by the way, Figure 4-21 does not represent the default state of this dialog form. Read the sidebar Tip, and I think that you'll understand why it was in this state for my screenshot. Let's bypass that for the time being.

You may have noticed that the radio button ◉ **dBase Expression result-based** is selected in the default configuration in your GoldMIne, and that the developers have included the **Expression: trim(Key1)**, refer to Figure 4-22. I would like to mention that this is really not a very relevant expression for this example. I think that I would have chosen one that included the use of the iif() function like:

```
iif(upper(trim(Contact1->Key1))==[CAR], [Car],
iif(upper(trim(Contact1->Key1))==[PROSPECT], [Prospect], iif(upper(trim(Contact1->Key1))==[SERVICE CENTER], [Service Center], [Other])))
```

However, before I went through all of that trouble, I think that I would just have selected the first option: O **Field value-based** which is more appropriate for the type of iif() function that I utilized. I would suggest that the dBase expression is best used for compound iif() expressions. Figure 4-23, on the next page, is the dialog form that is the result of having selected the O **Field value-based** option. You can see that when you don't require compound expressions, and the record type is based on a single field value that this is probably the best approach.

You'll notice that the Reule Definition for Record Type diloag form is a little discontinuous here. You get the statement: *"First, please choose the field to evaluate in this rule. Next, choose the value required for the rule to be applied."*, but then a set of radio buttons are thrown in between this statement and the **Field Name:** selection option. Well this is all new as of GoldMine Premium 8.5.1.12, and the developers just choose to toss it on the dialog form willy nilly. So let's continue on with this section first.

WARNING

*I would warn you that if you are making any changes through the **Record Types Administration Center** that you should have everyone close GoldMine, and then restart GoldMine after you have made your changes.*

Record type for new company/contact: is the leader for this radio button selection group, and your choices are:

- ○ **Use a default record type when creating new records**
- ◉ **When creating a new record, automatically use the current record type in view**
- ○ **Set record type manually when creating new records**

As always, the Help files are lacking and fail to help us understand these options so I'll just give you my best interpretation of what I am reading.

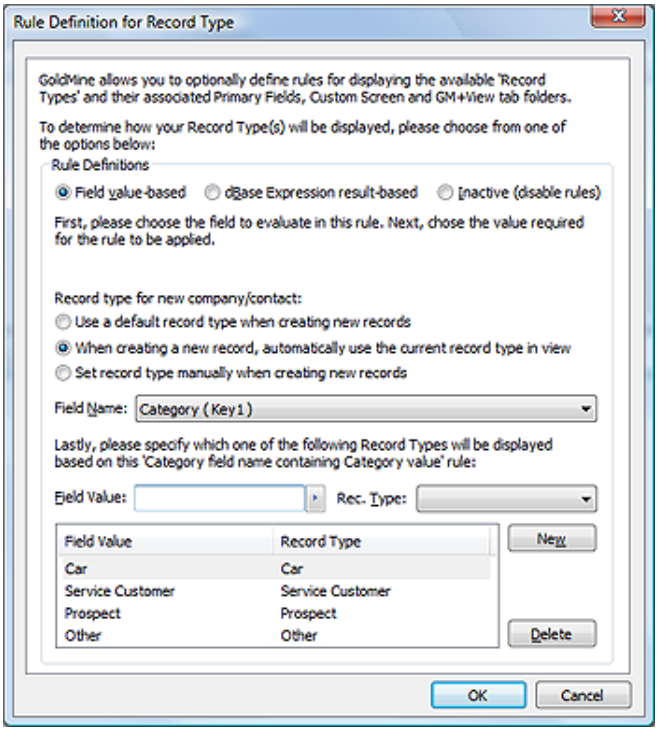

Figure 4-23

First, ○ **Use a default record type when creating new records** actual means that GoldMine will use **the** default record type, as defined by you earlier, as the record type for all newly created records.

Your second option, ◉ **When creating a new record, automatically use the current record type in view** is actual functioning as expected. What ever the record type for the active GoldMine Contact record is, when you create a New Contact record, will be the record type of the new contact record.

And, lastly, ○ **Set record type manually when creating new records** permits you to select the record type via the **New Record** dialog form **Record Type:** drop list which defaults to **[Plain Contact Record]**. As I see it, this is unnecessary as the GoldMine **File** | **New** ► menu offers one option for creating records of your various record types once you have selected to use ○ **Field value-based** for your rule. In the default state my GoldMine menu shows:

New Record

New Default Record Type...
New Car Type...
New Other Record...
New Prospect Record...
New Service Center Record...

Well then, now we appear to be back on track with the order of this dialog form so let's refresh ourselves on that previous statement again: *"First, please choose the field to evaluate in this rule. Next, choose the value required for the rule to be applied."*. You must first choose the field that will contain the value for the rule to be evaluated itself against. Since there is a finite set of fields the **Field Name:** value is selected from that finite list, and you are not permitted to input this information directly by hand. After you have selected the field name, you may add as many possible **Field Value:/Rec. Type:** value pairs as you wish by selecting the **New** button. You will notice in the default GoldMine configuration that the developers have included four possible value pairs.

This pretty much concludes the **Record Typing** section of this chapter, and for that matter, it pretty much wraps up the entire chapter as well. I would like to state, again for the record, that in the past the **Record Level Record Typing** feature of GoldMine has not been the most predictable feature with GoldMine. With each new version & build of GoldMine it has matured impressively. If working properly, **Record Typing** could enhance your database usage tremendously. Personally, I still utilize **Field Level Record Typing** for all of my needs as my industry does not require the reuse of fields in tables. It is true that **Field Level Record Typing** takes a lot more work, however, I have found it to be more stable, in the past at least, than **Record Level Record Typing**. I'd be most interested in hearing your opinions on this.