

Data Conditioning

While some companies are happy to store Contact data exactly as it's given to them, other companies prefer to apply a standard set of rules to some of that data, thus "conditioning" it. Typically, these rules involve Address fields, and the conversion of abbreviations to full words (or vice-versa); replacing certain words with others; and the application of "Proper Case" (capitalizing the first letter of each word in a data field).

Other rules, such as specifying the order in which some words appear in a field, may be desirable; but only the above three kinds of rules are readily obtainable via GoldBox. In fact, as we'll see, there are challenges even with them.

The application of Proper Case is not a problem; it just involves using a function. It must be noted, however, that Proper Case may not always give us what we want. If we want the name **McGill** to be capitalized as shown, Proper Case simply won't do it. But there are other things we can do to get there (discussion later).

On the other hand, word or abbreviation **replacement** can get quite tricky. GoldBox does have a table-based function called **GXTranslate** that will, if properly configured, evaluate each word (or word fragment) in the field to which it is applied. If a word/fragment is found in a certain column of a table that we control, then it is replaced by the value found in another column of that table. Here's an example:

123 21st St.

We can configure our function and table to replace St. with "Street", to give us:

123 21st Street

But consider this street address:

123 21 st St.

The space between "21" and "st" shouldn't be there, of course. But it is there, and it presents a problem. Replacement technology will need to replace any version of "st" (any case, and with or without a following period) that appears in an Address field with "Street". That would give us:

123 21 Street Street

Here's another one:

123 St. Augustine Rd.

What we really want is

123 Saint Augustine Road

but we won't get there with standard replacement technology. Instead, we'd get:

123 Street Augustine Road

Now, you might "imagineer" some additional rules, like "if 'St' follows a string of numerals, make it 'Saint'". Or, if it precedes a word/fragment that means "street" (like Avenue, or Rd.), then make it 'Saint'". But even if such rules were applicable within the context of GoldBox (they may be possible, but are **not** practical), there would always be exceptions to them, anyway.

What's needed is a strategy that recognizes some realities, and works around them. We need to:

1. Allow for the fact that some records will contain data that **simply cannot be automatically conditioned** to our satisfaction by any set of rules we can devise. So, we need to provide a way for humans to enter that data; and a way to ensure that GoldBox will never change that data during an Update (use a UDF to indicate such protection).

You might even expand on this idea and use Lookup.ini to protect certain fields from being changed even within GoldMine, for records that have that UDF marked.

2. For **routine data conditioning**, a list within the previously mentioned table would be specially configured to provide for replacing abbreviations or words or word fragments with the desired verbiage.

Some tough choices would have to be made, though. We cannot handle both St. (meaning "Street") and St. (meaning "Saint") in the same table, or with the same function. In this case, the choice would be to go with "Street", since it is far more common.

That means that there would need to be a sort of filter in the GXI¹ expression (using either a separate GXTranslate list, or perhaps a GX_Long_Expression) that would test the Address fields for the presence of a "Saint name". If one existed, the data would be carried through exactly as found in the Source file. Otherwise, the GXTranslate version of the Address data would go into GXI, and ultimately into GoldMine (except for records where the field is protected). Any record where translation was by-passed due to the presence of a "Saint name" would get a special UDF filled, and go into a special Group for review.

3. Notwithstanding item 1 above, it may sometimes be of interest to us when the data in a field that is thus protected differs in a significant way from the corresponding data in an Import Source file. For example, suppose we are processing web form data and a customer whose Address fields are protected per item 1 enters a completely new address (they've moved).

In a situation like that, we need to be able to test the Address fields in GoldMine against the Address fields in GXI, but in a stripped-down way. For this purpose, we'd want the common words and abbreviations (like Street and Apt.) to simply drop out before the comparison is made, so all we compare are the most significant words. Then, if a difference is detected, we can fill a UDF field that will ultimately result in the Contact going into another Group that gets further review.

So, how would we accomplish this "stripped down" evaluation? We'd use a **different list** within the same table (GoldBox provides for that), and in this list, all the common words and abbreviations would be replaced by a period. (GoldBox strips periods with GXTranslate, so it's really replaced with nothing).

4. Finally, we do occasionally use Address1 in matching schemes. Actually, what is normally done is to apply the SmoothComp function to Address1, specifying the desired list (one of the same lists we've already discussed). As it turns out, this is not a problem at all. In fact, it doesn't really even matter which of the two lists you use (although I tend to want to use the one with all the periods).

And what about my earlier comment about words like **McGill**? The first line of defense against problems like that is the ability to manually enter data the way you want it, and then to protect field contents from being over-written. But it's also possible to add such words to the main **GXTranslate** list for Addresses (the one without all the periods). Put **Mcgill** in the "**Search for This**" column, and **McGill** in the "**If found, return THIS**" column, and you'll get **McGill** every time. You must, of course, put your Proper function (if you use it) **INSIDE** the **GXTranslate** function.

While this is intended as a guide for designing a data conditioning system, it's important to keep in mind that each company may have specific challenges, and may require special procedures not mentioned here. Feel free to contact me for consulting, if needed.

¹ **GXI** refers to the mapped data in an Import/Update, including any data that is derived from a Mapped expression. **Proper(Source->Company)** is an example of a Mapped expression; it takes whatever data is in the Source->Contact field and converts it to Proper Case.