Bristol Group

Business Broker Searches

4/13/2010

Aaron Oyler
ISYS 540

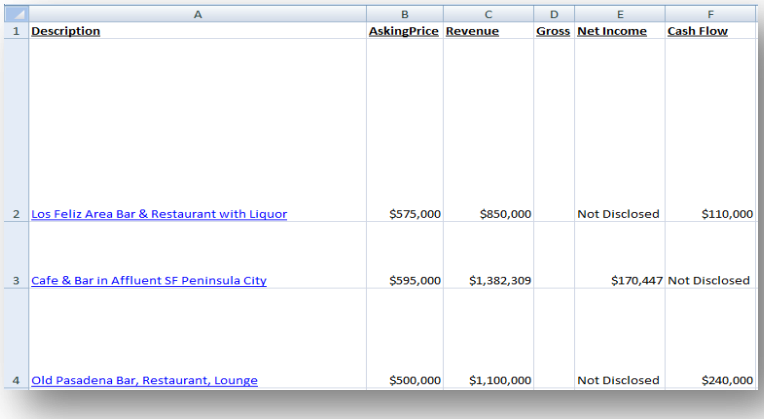
Problem Summary

The first time I bought a home I went to a realtor for help. The reason was simple. Using the MLS directory, the realtor had easy access to information about every home available for sale in the area. With this information and a basic understanding of my needs, a realtor could **quickly** narrow my search down to a few homes which saved a lot of time and increased my chances of finding the home that I wanted.

I have an acquaintance that is a business broker and he tells me that unlike for the sale of homes, there is not a single place to go for one stop shopping to buy a business. If what the broker has available for sale does not meet the buyer's requirements, then one of the parties must begin the long and tedious process of finding a suitable business. Essentially this boils down to visiting multiple websites that advertise businesses for sale, entering specific criteria, running a search, and manually sifting through the results. If a business looks promising the broker will do some simple financial calculations to further verify that the business meets minimum thresholds for consideration.

Proposed Solution

The purpose of this project is to make the broker's search more efficient by providing a single entry point for the business search from Excel. This will not take the place of existing web sites. From the list of results the user will be able to click on a hyperlink that takes them to the actual listing on the web.



	A	B	C	D	E	F
1	Description	AskingPrice	Revenue	Gross	Net Income	Cash Flow
2	Los Feliz Area Bar & Restaurant with Liquor	$575,000	$850,000		Not Disclosed	$110,000
3	Cafe & Bar in Affluent SF Peninsula City	$595,000	$1,382,309		$170,447	Not Disclosed
4	Old Pasadena Bar, Restaurant, Lounge	$500,000	$1,100,000		Not Disclosed	$240,000

Additional requirements / benefits of this solution are listed below.

- Simple user interface that provides a single point of entry for searches
- Broaden search and pull data from the top websites as defined by the business broker
- Automatically calculate financial ratios and display results in Excel
- Hyperlinks back to the actual listing on the web for additional data
- Configurable settings to filter data to be displayed

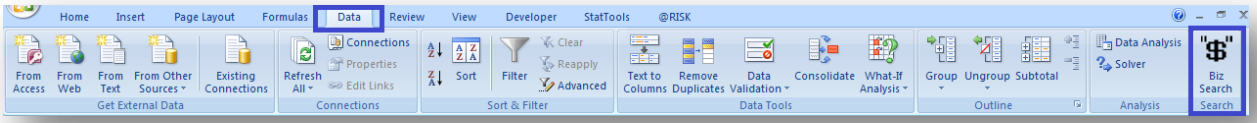
User Interface

The user interface will be very simple and made up of the following 3 components.

1) A “Biz Search” button on the “Data” ribbon
2) A business criteria search window
3) A settings window to control data output

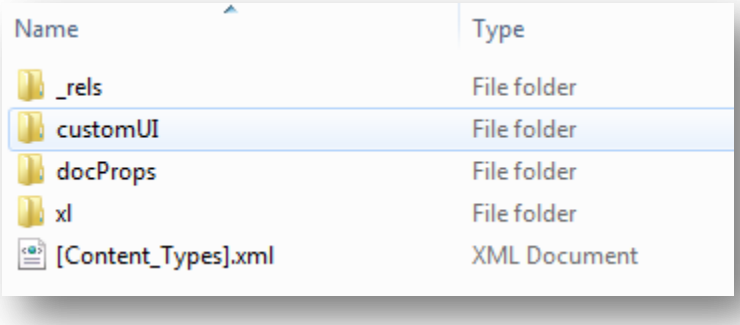
“Biz Search” Button

My intention was to make this functionality appear like an integrated component built specifically for use in Excel. Because the funtionality will go out and get external data off of the web it seemed like the best place for the new button was on the Data ribbon as part of the “Get External Data” group. I tried for hours to get this to work but eventually came to the conclusion that without going through a lot of work you can only add buttons to a new group at the end of the list. Although I wanted it in the “Get External Data” group it was not worth the extra effort so I just added it to the end.



There is a GUI application available for adding new buttons to an Excel ribbon but I could not get it to work so I added the button manually. The process for doing so is listed below.

1) Change the file extension for the workbook from xlsm to zip. All Microsoft Excel workbooks are nothing more than a zipped collection of xml files. These files can be viewed after changing the extension to .zip and then opening the folder.



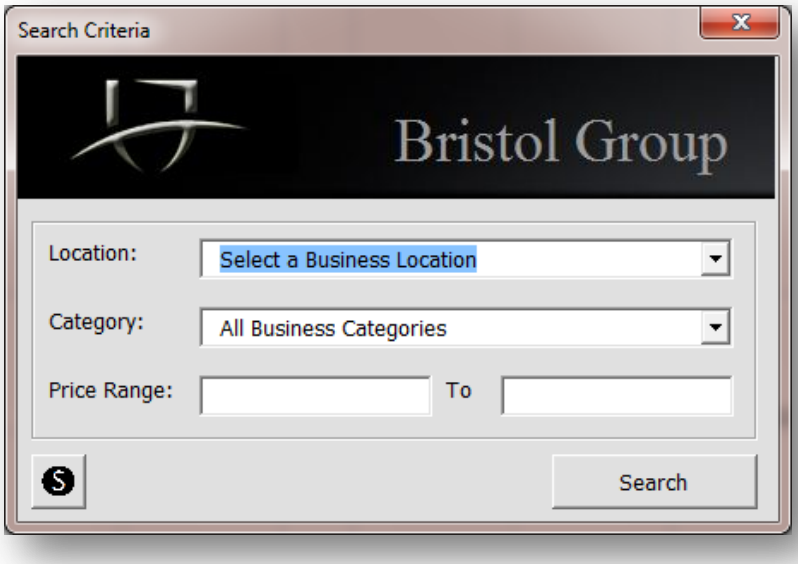
2) Navigate to, or create if necessary, the customUI folder.

3) Within the customUI folder, create a file named customUI.xml with the XML shown below.

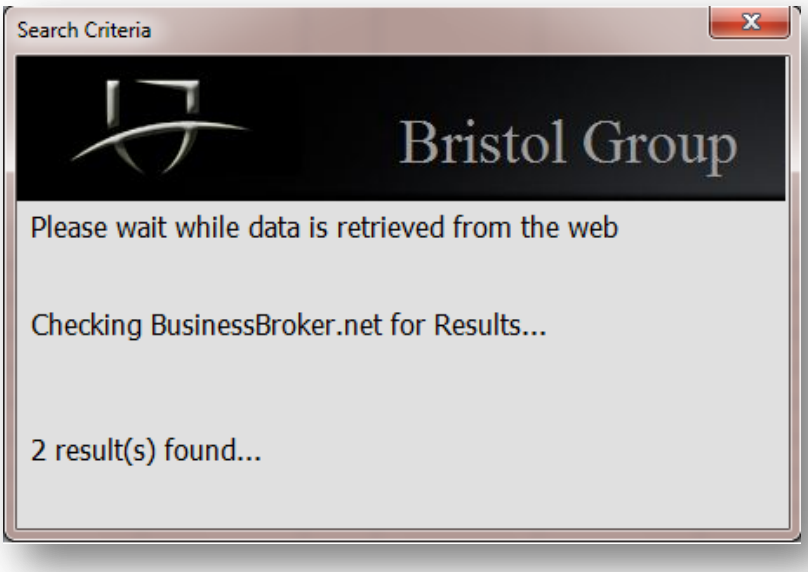
```
- <customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  - <ribbon>
    - <tabs>
      - <tab idMso="TabData">
        - <group id="GroupSearch" label="Search">
            <button id="btnBizSearch" label="Biz Search" size="large"
              onAction="Search4Business" imageMso="DollarSign" />
          </group>
        </tab>
      </tabs>
    </ribbon>
  </customUI>
```

Business Criteria Search Window

Selecting the “Biz Search” button calls the Search4Business routine which displays the search user form modally. After selecting a location, a category, and an optional price range the user can select the “Search” button to begin the process of searching for businesses that meet the specified criteria.

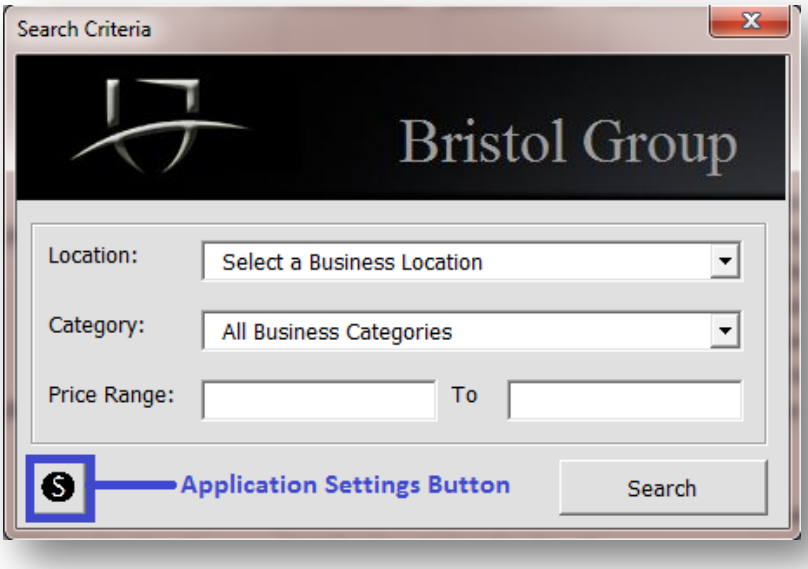


After the “Search” button is selected the window changes to display pertinent status information. This information is continuously updated throughout the search process, which can be lengthy depending on the chosen criteria. It shows information like which site is currently being checked and how many businesses have been found that match the user defined criteria.



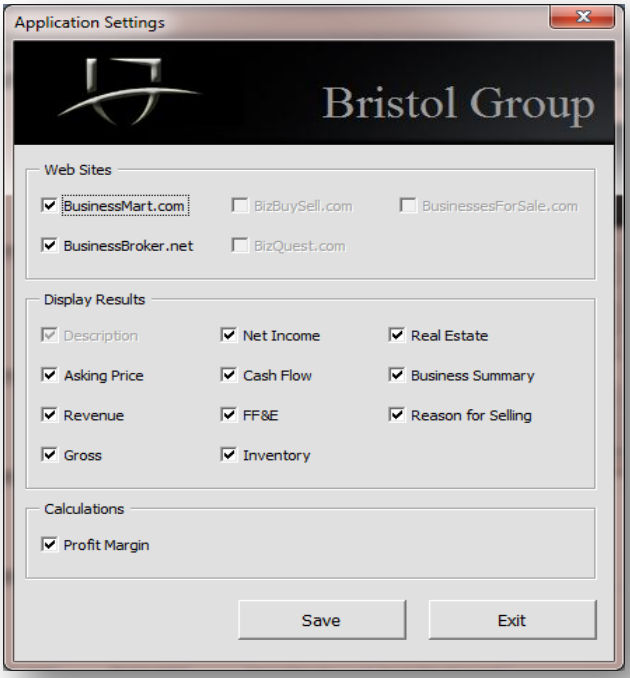
Results Settings Window

Users can control which sites are checked, what data is returned, and which calculations are automatically done via the settings option. This window is launched by clicking the button highlighted below on the criteria search window.



Once this button is selected a new window pops up that allows users to select what information is displayed on the results sheet. Note that only 2 of the 5 web sites are enabled. This is for 2 reasons. One, I ran out of time and two, I hope my business partner will like what I have done

and request the remaining 3 sites be enabled. Of course this would require some consulting revenue ☺ The options shown below are self explanatory and do not require further explanation.



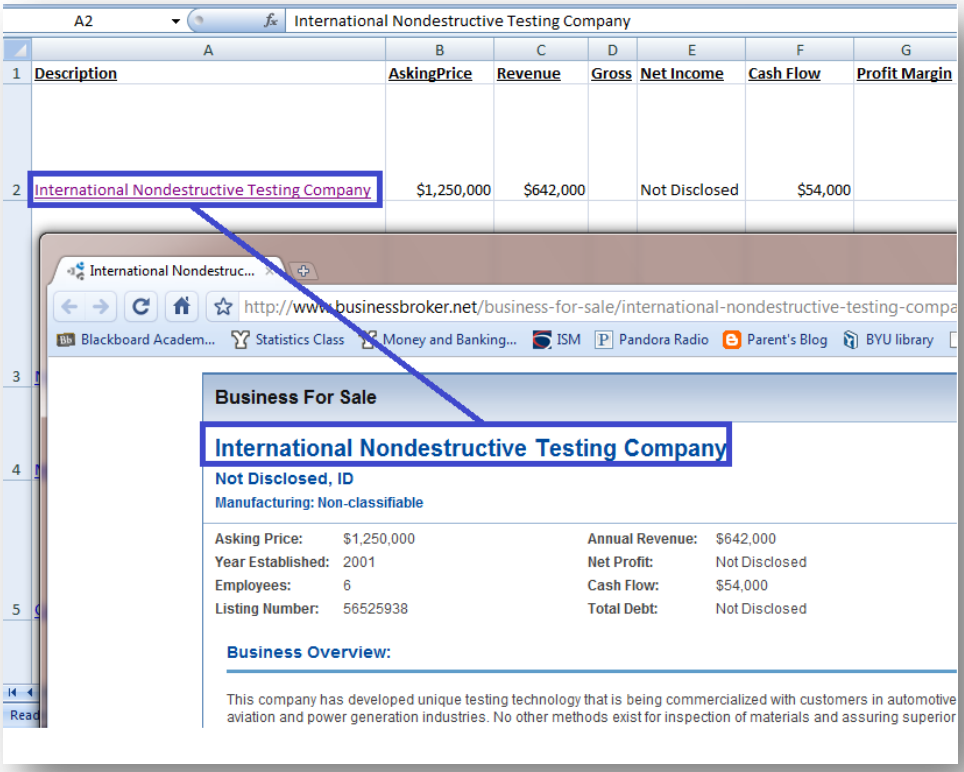
The first time the application is used all of the options are automatically selected. If a change is ever made, it is saved to the registry for future use. Rather than write the code to read and write to the registry, I went out on the web a found an existing VBA module that does this work. It was free and it works well. The entire module (modRegistry) is pretty large but I only use a couple of the routines which represent a small portion of my overall project. The code below is an example of how values are initialized when the form is first loaded. "UseBusinessMart" is a global variable that controls whether or not the BusinessMart.com website is used during each search. A similar variable is checked for the other 4 sites and for each display checkbox to see whether or not it should be selected.

```
If (RegistryKeyExists(HKEY_CURRENT_USER, "Software\Excel\Settings", True) = True) Then
    If (RegistryValueExists(HKEY_CURRENT_USER, "Software\Excel\Settings", "UseBusinessMart", True) = True)
        lsTemp = ""
        lsTemp = RegistryGetValue(HKEY_CURRENT_USER, "Software\Excel\Settings", "UseBusinessMart")
        UseBusinessMart = IIf(lsTemp = "0", "True", lsTemp)
    End If
End If
```

For the function RegistryKeyExists above, the 3rd parameter is an optional Boolean that when set to True creates the key if it does not already exist. The same is true for the function RegistryValueExists and the 4th parameter.

Example Results

Once the search is complete the results are added to a new sheet and only the values selected in the settings option are displayed. The main benefit of this output is that the user will be able to quickly scan the results for businesses to further evaluate. The user can click on the description hyperlink to go to the actual listing on the web and get additional details like the terms and conditions being offered by the seller.



Problems and Solutions

At the start of this project the following list of issues to be solved was generated.

1) How to reconcile the different search criteria for each site
2) How to standardize the different results data each site displays
3) How to efficiently navigate each site since they all use different HTML formats

Different Search Criteria

The first problem that I had to deal with was how to use a single set of search criteria across multiple sites that accept different values with varying levels of specificity. The table below shows each of the sites and a description of what search criteria they accept.

Web Site	State	City	Category	Price Range	Owner Financing	Keyword
BusinessMart.com The Business Search Engine	YES	NO	YES	NO	NO	NO
BusinessBroker.net Connecting Business Buyers and Sellers	YES	YES	YES	YES	YES	YES
BizQuest	YES+	NO	YES	YES	NO	YES
BusinessForSale.com Find Businesses Now	NO	NO	NO	NO	NO	YES
BizBuySell	YES	YES	YES	YES	NO	NO

BusinessForSale.com does not have good search capabilities so I did not consider it when making my decisions. In the end I decided to use the highlighted columns displayed above because they are standard across all sites. Price range is an exception for BusinessMart.com but it was easy to add a filter for price after the results were retrieved from that site. See below for the code.

```
If (Len(msAskingPrice) > 0) Then

    If (Len(txtBegin.Text) > 0) Then
        If (IsNumeric(txtBegin.Text) = True) Then

            If (CLng(msAskingPrice) < CLng(txtBegin.Text)) Then
                lbAdd = False
            End If

        End If
    End If

    If (Len(txtEnd.Text) > 0) Then
        If (IsNumeric(txtEnd.Text) = True) Then

            If (CLng(msAskingPrice) > CLng(txtEnd.Text)) Then
                lbAdd = False
            End If

        End If
    End If

End If
```

The next step was to “translate” the specified input into the appropriate value. For example, if I want to search for businesses in Idaho, the value “Idaho” needs to be “translated” into the appropriate numeric value to be assigned to the HTML control for each site. No two sites are the same.

```
lsCategory = TranslateBusinessMartCategory(cmbCategory.Value)
lsLocation = TranslateBusinessMartLocation(cmbLocation.Value)

'*************************************************
'** Check to see if category exists for this site
'*************************************************
If (lsCategory <> "-999") Then

    IE.Document.all("bloc").Value = lsLocation
    IE.Document.all("pricat").Value = lsCategory

    IE.Document.forms("busform").Submit
```

Each site has its own translation procedure which is basically a list of hardcoded locations and the appropriate values. If I had more time to work on this project I would have put these translation values into a database instead of hard coding them.

```
Public Function TranslateBusinessMartLocation(ByVal psValue As String) As String

    Select Case psValue

        Case "Select a Location", "SEARCH ENTIRE US"
            TranslateBusinessMartLocation = "%"
        Case "Alabama"
            TranslateBusinessMartLocation = "1"
        Case "Alaska"
            TranslateBusinessMartLocation = "2"
        Case "Arizona"
            TranslateBusinessMartLocation = "3"
        Case "Arkansas"
            TranslateBusinessMartLocation = "4"
        Case "California"
```

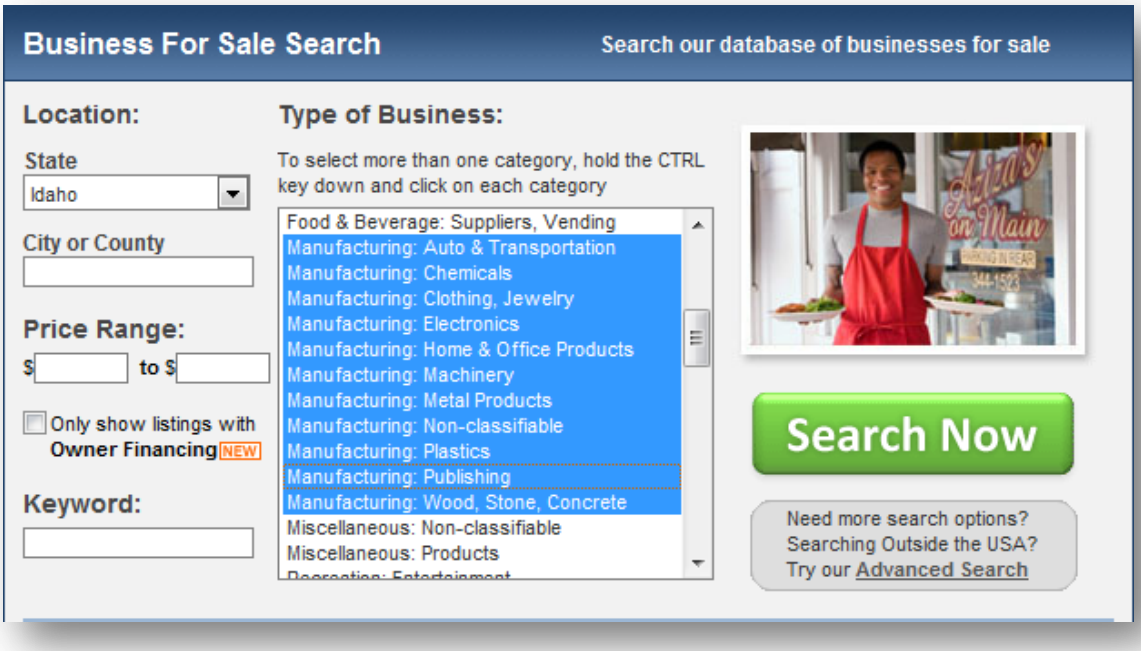
The last issue in this section was that some sites have more generic search categories than others. For example, BusinessMart.com has a single category for manufacturing businesses, but BusinessBroker.net has it broken up into multiple categories. The drop down list on the user form contains the most generic value and then when it is translated for the more specific site, all options are returned and selected. I have included the code below and an example of the select box on the BusinessBroker.net web site. This web site is not visible during the search.

```
lasValues = ParseString(lsCategory)

For Each Thing1 In IE.Document.all("ListboxIndustry").Options
  Thing1.Selected = False
Next

For Each Thing2 In lasValues
    For Each Thing1 In IE.Document.all("ListboxIndustry").Options
      If Thing1.Value = Thing2 Then
        Thing1.Selected = True
        Exit For
      End If
    Next
Next

IE.Document.all("bfs_SearchNow").Click
```



Different Output Data

Each site returns a lot of the same data with a few minor exceptions. Rather than try to deal with merging differences, I just make all critical data available and the user can decide what they want to see and what they don't. At first I was going to add everything to the sheet and hide columns but in the end I decided to do it a little differently. There are two global variables for each potential column. One for whether or not it will be displayed and one for the actual

column it will be in. When it comes time to add a new data row I just reference the data columns that were populated in the routine below.

```
Public Sub UpdateDisplayColumns()
Dim liCount As Integer

    ShowDescriptionCol = ""
    ShowAskingPriceCol = ""
    ShowRevenueCol = ""
    ShowGrossCol = ""
    ShowNetIncomeCol = ""
    ShowCashFlowCol = ""
    CalculateProfitMarginCol = ""
    ShowFFECol = ""
    ShowInventoryCol = ""
    ShowRealEstateCol = ""
    ShowBusinessSummaryCol = ""
    ShowReasonForSellingCol = ""

    If (ShowDescription = True) Then
        liCount = liCount + 1
        ShowDescriptionCol = Convert2Column(liCount)
    End If
    If (ShowAskingPrice = True) Then
        liCount = liCount + 1
        ShowAskingPriceCol = Convert2Column(liCount)
    End If
    If (ShowRevenue = True) Then
        liCount = liCount + 1
        ShowRevenueCol = Convert2Column(liCount)
    End If
```

Nonstandard HTML Searches

I was hoping to create a single standard routine for searching for businesses on each website but I soon realized that would not be possible. Each site is different in what controls they use and how they display their results. In the end I had to create a separate routine for each site. Creating the first routine was helpful for all of the rest because it established the pattern I would use from then on out and essentially all that I had to change was the syntax.

One thing that was very helpful was to create a class named “TextParser” that I could use to search a text string for specific data. This was essentially a duplication of what was done in class except for one piece of functionality. I added an optional parameter to the FindNext routine named psBeforeText. This made it so that I could locate a table within the HTML source code and then create a loop to return all of the <TR> rows before the next </TABLE> tag and not have to worry about returning every <TR> tag in the code.

```
'************************************************************
'** Check to see if we need to limit the end of the search
'************************************************************
If (Len(psBeforeText) > 0) Then

    llEnd = InStr(llPosition, psSearchString, psBeforeText)

    If (llEnd > 0) Then

        lsSaveString = psSearchString

        psSearchString = Left(lsSaveString, llEnd)

    End If

End If
```