ERP apps

thoughts on improving your business by xkzero

How to Prevent Duplicate Customer Names with Scripting

Posted on **April 27, 2015** by **Alnoor Cassim**

Scripting Tips for Sage 100 ERP

Blogger: Alnoor Cassim, xkzero Technical Services

Sometimes, our software does TOO much and we want to intervene with a script to *prevent* an action from occurring. Here is an example of such a scenario:

— **Sometimes a script is just what you need to block an action!**

When a new customer is entered in Customer Maintenance, you want the following steps to occur:

1. Check the customer name against all other records for duplicates.
2. Prevent any duplicate customer names from being repeated on the new customer record.
3. Display a message to indicate that the name is already being used. Include the customer number.
4. Audibly reinforce the message with an alert sound or spoken words.

Is it possible to create these steps with scripting? Most certainly!
Note that a well thought-out and researched script can accomplish the task in an efficient and optimal way. However, a less carefully planned script (or modification) can run into performance issues and may not return the correct results 100% of the time.

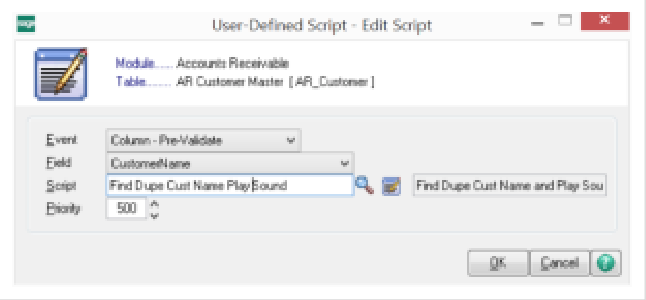
The good news is you will see the well-designed version! It is a short script but appears to be longer because it is heavily commented. Bonus: Many concepts applied in this script can be also be applied to other scripts you write.

What kind of script do you need to create?
Create an *event script* (a.k.a. *User Defined Script*) that runs when the customer name is entered in Customer Maintenance. The event type we are using is called *Column Pre-Validate*. All of the “pre” events (pre-validate, pre-write, pre-delete) are designed for you to add a user-defined validation and *prevent an action from occurring*.

How do you create this script?

Follow these steps:

1. Go to the Custom Office/Main Menu, and make these selections:
   “User-Defined Field and Table Maintenance”
2. Under the “Accounts Receivable” heading, choose: “AR Customer Master.”
3. Right-click and choose “User Defined Scripts.”
4. In the “User Defined Script” window, click the “Add” button. This will open the “Add Script” window.
5. In the “User-Defined Script–Add Script” window, as shown below, make these choices:
   Event: “Column–Pre-Validate”
   Type in a name for your script, such as this:
   “Find Dupe Cust Name Play Sound”



Module...... Accounts Receivable
Table....... AR Customer Master [AR_Customer]

Event: Column - Pre-Validate
Field: CustomerName
Script: Find Dupe Cust Name Play Sound | Find Dupe Cust Name and Play Sou
Priority: 500

6. You will see a message indicating that the script file does not exist, followed by a question, “Do you want to create it? Yes/No?”
   Yes, you do want to create the script! Click: “Yes.”

7. Now the “Edit Script” window will appear. Copy the following code and paste it into your script editor window. **Note: It is important to use the code *exactly* as shown below, including line breaks and spaces**.

```
'Alnoor Cassim - xkzero - Prevent Duplicate Customer Names and Play Sound

'Init VARs
DupeCustNo = "" : DupeCustName = ""

'Run only for new customers by verifying EditState = 2
'An EditState = 1 means an existing record, 0 means no record loaded.
'oBusObj is the primary connection to AR_Customer table.

If oBusObj.EditState = 2 Then

'Create a separate new connection to AR_Customer
'Use the service object since we are not writing back data
'Use this syntax

Set oCustomer = oSession.AsObject(oSession.GetObject("AR_Customer_svc"))

'Searching one record at a time from top to bottom would be slow
'Set the KNAME index as the browse index to do a fast search of Customer Name
'KNAME from File Layouts: CustomerName+ARDivisionNo+CustomerNo

retVal = oCustomer.SetBrowseIndex("KNAME", "")

'The variable called "value" represents what you typed in for the Customer Name.
'Assign "value" to your own variable. We will call our variable "Name".
'Strip any trailing spaces and force Upper Case to avoid a case sensitive search.

Name = UCase(RTrim(value))

'We do not have the values for all 3 columns of KNAME index.
'So create a filter to pass in just the CustomerName as a partial value.
'This is done with the SetBrowseFilter() function.

retVal = oCustomer.SetBrowseFilter(Name)

'Now this MoveFirst() command will both be filtered by name and use the KNAME index.
'This will result in a fast search!

retFound = oCustomer.MoveFirst()

If retFound = 1 Then

'retFound is the return value. It will hold either a 1 (found) or 0 (not found).
'If 1 then we found at least a partial match but now check if it is an exact dupe.
```

```
retVal = oCustomer.GetValue("CustomerNo$", DupeCustNo)
retVal = oCustomer.GetValue("CustomerName$", DupeCustName)

'A short dupe is OK. Ex: You typed "Road" and the search found "RoadRunner"
'Only reject if you typed in RoadRunner. This is how:

If Name = UCase(RTrim(DupeCustName)) Then

'The SetError() function will both show the message and prevent the dupe name.

sMsg = "This name is already used with Customer No " & DupeCustNo
retVal = oScript.SetError(sMsg)

'Play a Windows sound event using the ProvideX PRINT BEEP function.
'This plays back sound on the wkstn even in Sage 100 Advanced and Premium

oScript.Execute("PRINT "+"'BEEP'"+CHR(40)+CHR(34)+"*.Default,w"+CHR(34)+")")

'That was the registered default event sound in Windows.
'Look in registry for more registered sounds. Leave the "*" in front.
'HKEY_CURRENT_USERAppEventsEventLabels
'E.g. "*EmptyRecycleBin,w"

End If 'check if a duplicate name was found

End If 'allow for short duplicate names

End If 'check if on a new record
```

How does this script work?

To better understand the logic behind the script's functionality, read this:

"Init VARs"
In the first section, we "initialize" (init) the variables to prepare for use later in the script.

"IF/THEN/ELSE"
Through these stated conditions, we apply logic to *check if we're in a new record*, then later to *check if we have found a customer name match*, and then later to *check for an exact name match*.

How do we find out if the customer is new or existing?
We do this by checking the "edit state" as indicated by the following values:

2 = new customer

1 = existing customer
0 = no customer on the screen

How to Use a Secondary Customer Object
When the script starts we already have an existing Customer object in memory called **oBusObj**. For this script we need to create a secondary connection to the same Customer object by using the **GetObject()** function. We will call that second object **oCustomer**. In our case we set **oCustomer** as the handle for **AR_Customer_Svc**. However, if we need, to write back data we use **AR_Customer_bus** instead.

How to Use an Alternate Index for Searching, Set a Browse Filter, and Find the Result
The primary index is called **KPRIMARY** and contains the **ARDivisionNo** and **CustomerNo** columns. We can use **KPRIMARY** for finding the duplicates but would have to check the **CustomerName** column on every row until a match is found. This type of sequential top to bottom search is very slow.

Instead, we look for an index that starts off with **CustomerName**. From the File Layouts link in Sage 100 ERP, under **AR_Customer** table, we see the **KNAME** index contains what we need. We will apply this to the **oCustomer** handle. Also, to make sure we only change the index for the purpose of searching, we use the **SetBrowseIndex()** function with "**KNAME**" passed in as *1st argument* and an empty string as the *2nd argument*.

One issue with KNAME is that it comprises three columns that make up this alternate key, yet we only want to search by the one **CustomerName** column. So, we must set a partial key by creating a filter to pass in the **CustomerName**. This is done with the **SetBrowseFilter()** function and we apply it to the **oCustomer** handle as well.

Now we use the **MoveFirst()** function. Normally it would just move to the first row of the record set. In this case it finds the first matching result based on our browse filter and searches in KNAME order.

The **retFound variable** represents a return value, just like the **retVal** variable does. We can use any numeric variable for return values.

If our search finds a match, then **retFound = 1**.
Otherwise, **retFound = 0**.

Partial Duplicate vs Exact Match
After using **MoveFirst()** and after **retFound = 1**, we can have a situation where we have a partial duplicate match instead of an exact match.

For example, if you type "Grass" for the customer name, the search may return "GrassHopper Pesticide Corp." You don't want this search to prevent you from using "Grass," so to find an exact match, we do the **GetValue()** function to compare the actual name against the searched name.

Rejecting the Customer Name

Once an exact match has been found, now we have to actually reject/prevent the customer name that was entered in the screen from being used. This is accomplished with the **SetError("User Defined Message")** function. It will both show your user defined message and prevent the name you enter on the screen from being used.

Playing a Windows System Sound

Use the PRINT BEEP command noted in the script to play a Windows sound event. In Advanced and Premium environments the sound will play on the workstation.

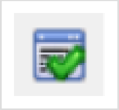
You're almost done!

Now you can accept the script.

This will return you to the "Add Script" window.

1. To close this screen and return to the "User Defined Scripts" window, click "OK."
2. Next, to go back to the main "User Defined Field" and "Table Maintenance" window, click "Close." Here, you must also click "Close" ***again*** to see the "Script Compile" window.
3. IMPORTANT: When the "Script Compile" window appears, ***click the "Compile" button***.

Note that you can further edit the script, if necessary. To do so, from the "Custom Office/Main" menu, choose the "User Defined Script Maintenance" task. When done editing the script, to syntax check the script, click the Check Script button:



To save the script, click the Accept button.

The next step is to click the Compile button in the lower-left of the screen. When the Script Compile window appears, click the Compile button again followed by the Close button.

How to Use the Windows Text To Speech (TTS) API to Speak the Words

As an alternative to playing a system sound you can play back the spoken words of the error message using a Windows Accessibility feature called Text To Speech (TTS). This is challenging to use in Sage 100 Advanced and Premium because the playback needs to occur at the workstation, not the server. *[Credit goes to Sage 100 VAR Vrakas/Blum for coming up with the TTS idea and asking how to play the voice on the workstation.]*

Sage 100 ERPAdvanced/Premium:
CHALLENGE & SOLUTION

It turns out running any event script for Advanced/Premium is a challenge, since by design it executes on the server yet we need it to execute on the client. Button scripts have a nice option to "Execute Script on Client" vs "Execute Script on Server" but event scripts do not.

This is when we need to pull out our bag of tricks.

We will need a total of 2 scripts. The 1st script is actually the same one as above but we substitute the PRINT BEEP code with the following script:

```
'Text To Speech version of Playing Sound

'Check if we are on Sage 100 Advanced or Premium edition
If oSession.CS Then

'Set location for Workstation Script. Change the file name as needed.
WkstnScript=oSession.PathCSRoot+"CMScriptDupeCustNamexkZ_PlaySoundWkstn.vbs"

WSH = "WScript.exe "  'This is the Windows Scripting Host
vbsCMD = CHR(34) & WSH & WkstnScript & CHR(34) 'Form the o/s cmd to run a VB Script
oScript.Execute("ShellCMD$ = " & vbsCMD) 'Pass the cmd into a pvx variable ShellCMD$

'Fire off the script at the workstation. The [wdx] indicates run from the client.
oScript.Execute("CALL "+CHR(34)+"[wdx]SYZDLL;SHELL_AND_WAIT"+CHR(34)+",ShellCMD$")

Else
'We are on Sage 100 Standard so it will already run from the client.

'Tap into TTS feature. More info can be found here: http://goo.gl/cM6dYJ
If IsObject(oVoice) = False Then
Set oVoice = CreateObject("SAPI.SpVoice")
End If

Words = "This customer name is already used."
retVal = oVoice.Speak(Words)

End If
```

The Text to Speech workstation script (for Advanced and Premium) will look like this:

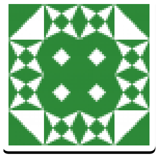
```
'Workstation Script but physically located on server here
'..Mas90CMScriptDupeCustNamexkZ_PlaySoundWkstn.vbs
'NOTE: Rename the file as needed but you must save it as VBS not TXT.

'Tap into TTS feature. More info can be found here: http://goo.gl/cM6dYJ
If IsObject(oVoice) = False Then
Set oVoice = CreateObject("SAPI.SpVoice")
End If

Words = "This customer name is already used."
```

```
    retVal = oVoice.Speak(Words)
```

This is the third of many Sage 100 ERP scripting examples we will share through the xkzero blog. If you need help with scripting or other programming or technical issues with your Sage ERP, no matter how complex, please feel free to contact us:

xkzero Technical Services
Email: info@xkzero.com
Call: 847-416-2009

This entry was posted in **Sage 100 ERP**, **xkzero Technical Services** and tagged **scripting**, **tips** by **Alnoor Cassim**. Bookmark the **permalink [http://erpappsblog.com/?p=1508]** .



About Alnoor Cassim

Director, xkzero Technical Services for Sage 100 ERP. Providing companies solutions to utilize and enhance their ERP software to grow their business, identify and create solutions that streamline business processes, improve productivity, increase profitability, and provide win-win scenarios for all.

View all posts by Alnoor Cassim →