Paper 111-2008

"I Shall RETURN" – Key Feedback in SAS/AF®

Greg McLean, Statistics Canada, Ottawa, Ontario, Canada

ABSTRACT

Often within our SAS/AF applications, there is a requirement for the user to enter text. However, as a default, the user must hit the "Return" or "Enter" key to execute the object's SCL (Source Component Language) code. Although this is the default behavior, it may be beneficial to trigger some other action or piece of SCL code based on each keystroke that the user types.

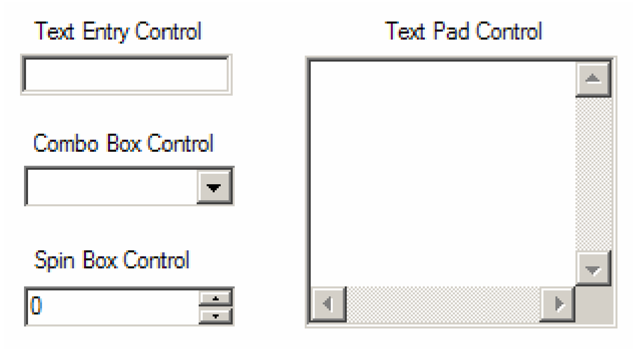
The use of "**Key Feedback**" in SAS/AF will improve the professional appearance as well as increase the productivity, acceptability and ease of use of a given SAS/AF Graphical User Interface.

This paper will illustrate how **Key Feedback** can be used in conjunction with several SAS/AF visual Objects within your GUI applications. The intended audience for this paper is SAS developers with a medium to advanced knowledge of SAS/AF and SCL software.

INTRODUCTION

When developing Graphical User Interfaces (GUIs) it is important to ensure that it is easy to use as well as intuitive. The type of controls used as well as how they function will have an important impact on productivity, acceptability and ease of use.

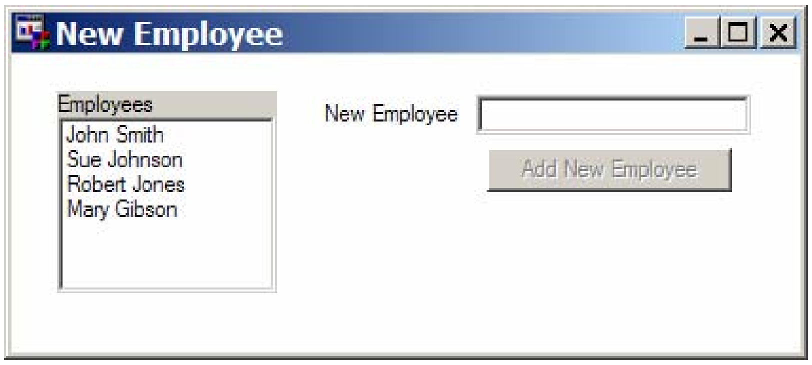
The use of **Key Feedback** can be utilized in such a way to improve clarity as well as professional appearance of your SAS/AF applications. The concept of **Key Feedback** is quite simple. It allows your GUI application the opportunity to run SCL code as each character is typed instead of after the "Return / Enter" key is pressed. **Key Feedback** can be used with the following visual controls:



Although there are many potential uses of **Key Feedback**, this paper will describe this concept using one particular scenario as an example.

EXAMPLE

Assume that we have a SAS/AF application that keeps track of current employees. One of the Frames within this application is used by the user to add new employees.

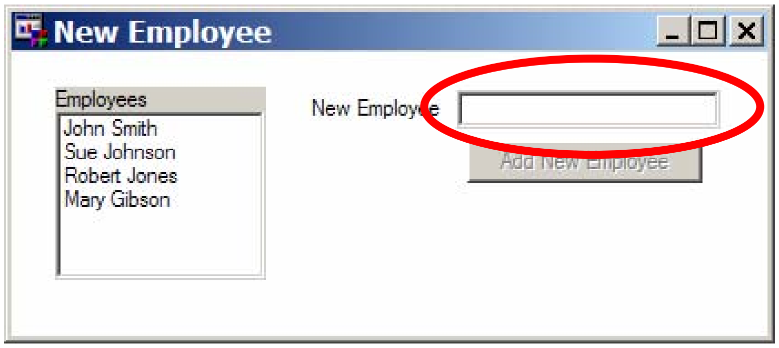


The “Add New Employee” pushbutton will be “Grayed” or “UnGrayed” depending on what is contained in the text entry field (new employee). The following table outlines the different states of the “Add New Employee” pushbutton based on the corresponding conditions.

Condition	State
❖ Text Entry Field is blank ❖ Text Entry Field contains a name that is already in the list	Pushbutton is GRAYED
❖ Text Entry Field contains text (name) that is **not** in the list	Pushbutton is UNGRAYED

PROBLEM
Using a traditional approach to add a new employee, the user would type in a name and press the “Add New Employee” pushbutton. However, he/she would not be certain if the employee already existed in the current list of employees, particularly if the list was quite long. In fact, using a traditional approach, verification of what the user has typed would not be verified or processed until after the “Return / Enter” key has been pressed. At that point the SCL program could verify the name and if required present a message to the user that the entered name already existed in the list. Also, what would happen if the user did not enter any text in the Text Entry Field yet hit the “Add New Employee” pushbutton? Again a message would be required to tell the user that a new name must be first entered.

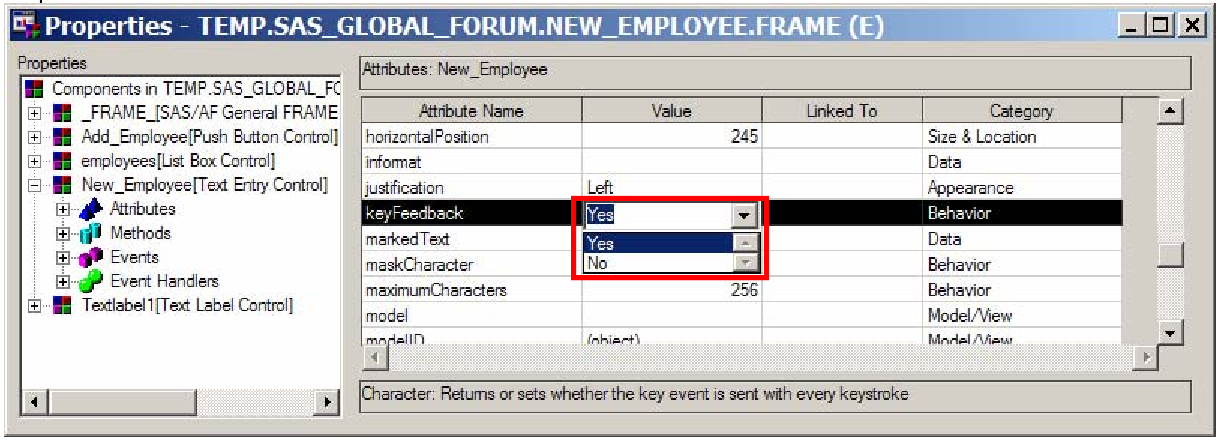
SOLUTION
Although the approach of using messages to notify the user of various errors or scenarios is feasible, the use of **Key Feedback** could be used instead to provide instant feedback to the user after each keystroke is made. This section will illustrate how we set up **Key Feedback** for the ***Text Entry Control Object*** (where user would type a new name).



John Smith
Sue Johnson
Robert Jones
Mary Gibson

1.) TURNING ON “KEYFEEDBACK” PROPERTY
The first thing that we must do is turn on the “**KeyFeedback**” property for the text entry object. This will allow an event to occur each time a keystroke is pressed. The “**KeyFeedback**” property can be set in two different ways.

Compile Time
At development / compile time, locate the “**KeyFeedback**” property for the text entry object. Select “Yes” from the dropdown selection list.



Components in TEMP.SAS_GLOBAL_FC
FRAME[SAS/AF General FRAME
Add_Employee[Push Button Control]
employees[List Box Control]
New_Employee[Text Entry Control]
Attributes
Methods
Events
Event Handlers
Textlabel1[Text Label Control]

Attributes: New_Employee

Attribute Name	Value	Linked To	Category
horizontalPosition	245		Size & Location
informat			Data
justification	Left		Appearance
keyFeedback	Yes		Behavior
markedText	Yes		Data
maskCharacter	No		Behavior
maximumCharacters	256		Behavior
model			Model/View
modelID	(object)		Model/View

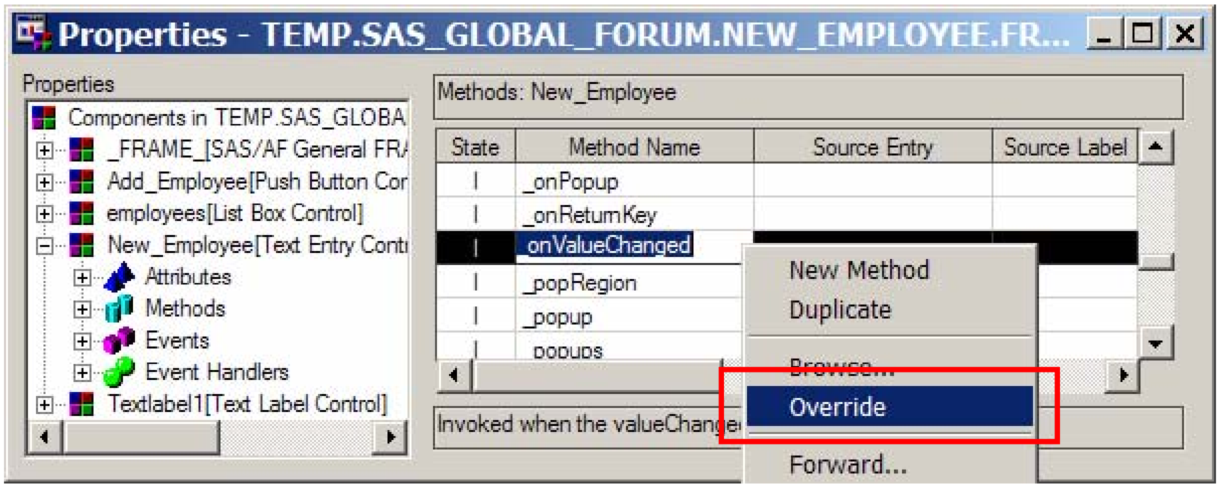
Character: Returns or sets whether the key event is sent with every keystroke

Run-Time
It is also possible to set the "**KeyFeedback**" property at run-time. The following code should be placed in the INIT section of the Frame SCL program:

```
       New_Employee.KeyFeedback = "Yes";
```

2.) OVERRIDING "_ONVALUECHANGED" METHOD
Next we must tell our application what to run when the user types in this text entry object. To do this we must override the "**_onValueChanged**" Method for the text entry object. There are two ways that this can be accomplished.

Compile Time
The first approach can be done at development / compile time. We must tell the text entry object that we wish to override the "**_onValueChanged**" method and tell it where to find the corresponding SCL code to run. To do this we go to the properties window for the text entry object and click on "Methods". Scroll through the list of "Methods" and find the "**_onValueChanged**" method. Using the secondary mouse button, click on the entry to select "*Override*" from the drop-down list.

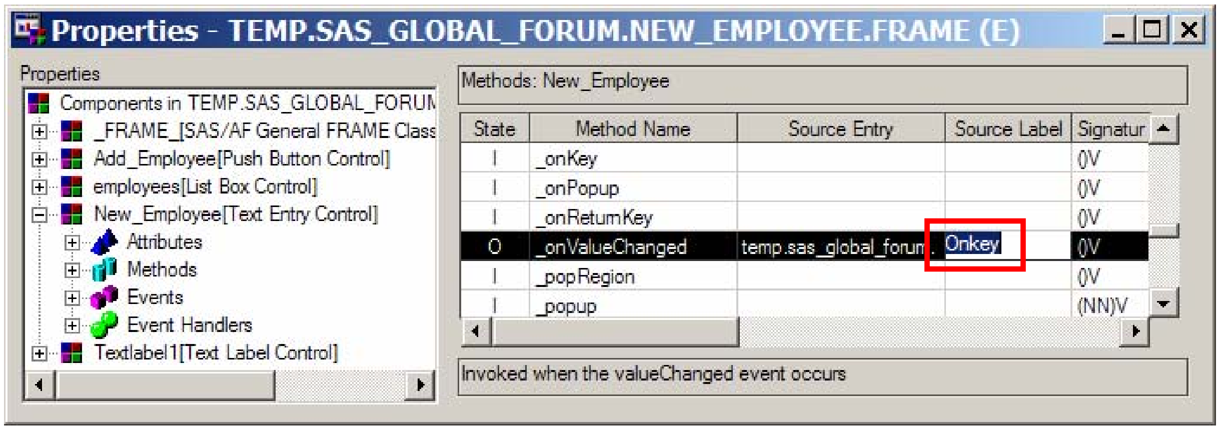

Properties - TEMP.SAS_GLOBAL_FORUM.NEW_EMPLOYEE.FR...
Properties
Components in TEMP.SAS_GLOBA
FRAME[SAS/AF General FR/
Add_Employee[Push Button Cor
employees[List Box Control]
New_Employee[Text Entry Contr
Attributes
Methods
Events
Event Handlers
Textlabel1[Text Label Control]
Methods: New_Employee
State
Method Name
Source Entry
Source Label
_onPopup
_onReturnKey
_onValueChanged
_popRegion
_popup
_popups
New Method
Duplicate
Browse...
Override
Forward...
Invoked when the valueChange

We then must specify a 4 level name of the entry, which contains the SCL code to be run when the user types a character in the text entry field. Thus a new SCL entry must be created. It is suggested that a separate SCL entry be created in your SAS Application Catalog that will contain all of your overridden methods.

Place the cursor in the "Source Entry" field for the "_onValueChanged" method or use ... beside the field to make or type your selection.

<SAS LIBREF>. <Catalog Name>. <Entry Name> .SCL

We then must specify the "Source Label" within the new SCL entry that is to be used or run. Because many methods or "Labels" can exist in a given SCL entry, we must be specific about the one to be used. Place the cursor in the "Source Label" field for the "**_onValueChanged**" method and type your "Label" name. In our example we will call it "Onkey".


Properties - TEMP.SAS_GLOBAL_FORUM.NEW_EMPLOYEE.FRAME (E)

Properties
- Components in TEMP.SAS_GLOBAL_FORUM
  - _FRAME_[SAS/AF General FRAME Class
  - Add_Employee[Push Button Control]
  - employees[List Box Control]
  - New_Employee[Text Entry Control]
    - Attributes
    - Methods
    - Events
    - Event Handlers
  - Textlabel1[Text Label Control]

Methods: New_Employee

State	Method Name	Source Entry	Source Label	Signatur
I	_onKey			()V
I	_onPopup			()V
I	_onReturnKey			()V
O	_onValueChanged	temp.sas_global_forum.	Onkey	()V
I	_popRegion			()V
I	_popup			(NN)V

Invoked when the valueChanged event occurs

Run-Time
The second way that we can override the "**_onValueChanged**" method is at run-time in the SCL code. The following code would be placed in the INIT section of the program:

```
        New_Employee._setinstancemethod('_onValueChanged',
                                        'TEMP.SAS_GLOBAL_FORUM.METHODS.SCL',
                                        'onkey');
```

Note: The second parameter would be specific to the location (4 level name) of your SCL entry that contains the new SCL method code to be run when the users types a character in the "New_Employee" object.

3.) NEW SCL METHOD
And finally we would have to create code in the new SCL entry that would process each time the user types a keystroke in the text entry field (new employee). Again, there are several approaches that could be used. We have chosen to keep it simple. The SCL code for the overridden method "**_onValueChanged**" merely redirects the focus back to the Frame SCL program. More specifically, to run the object label of the text entry object.

The following Method (SCL code) would be placed in the new SCL entry located in our Application Catalog. As previously suggested, this SCL entry could contain this overridden method along with other overridden methods. In our example we have named this SCL entry: "METHODS.SCL".

```
        _SELF_ = _SELF_;
        onkey:
           METHOD;
           _SELF_._objectLabel();
           ENDMETHOD;
```

At runtime the code illustrated above would be executed after each keystroke in the text entry field. This code actually directs control back to the object label ("New_Employee:") in the original Frame SCL program.

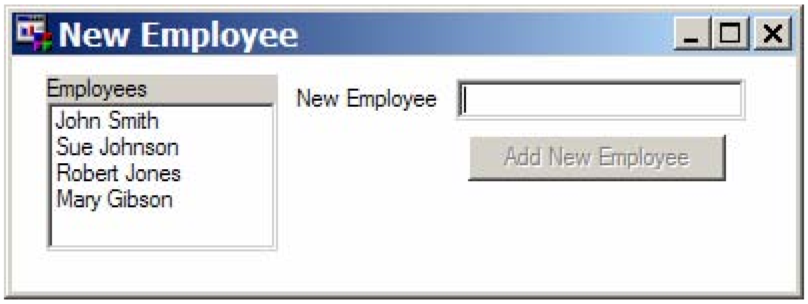
4.) EXAMPLE SCL CODE
And finally we need to add SCL code to the text entry object label section that will test the various conditions and set the “Add New Employee” to Gray or UnGray.

```
NEW_EMPLOYEE:
   if (new_employee.text = " ") then add_employee._GRAY();
   else do;
      pos = SEARCHC(employee_list,new_employee.text);
      if (pos > 0) then add_employee._GRAY();
      else              add_employee._UNGRAY();
   end;
RETURN;
```

Note: In our example we store all of our current employees in a SCL list.

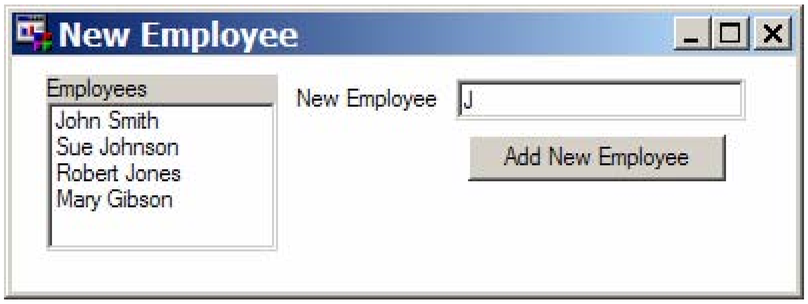
5.) RESULTS
Now let’s examine four different scenarios that would have an effect to the visual appearance of the Frame.

1.) Start of “New Employee” Frame:



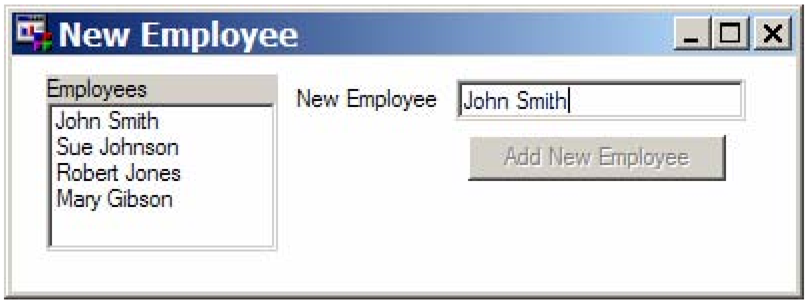
Note: Since there is no text in the “New Employee” field, the “Add New Employee” pushbutton is “Grayed” and cannot be selected.

2.) User types a character:



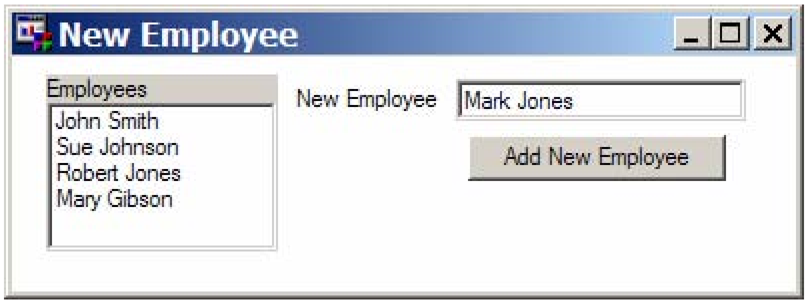
Note: Now that text appears in the “New Employee” text entry field, and does not exist in the “Employees List”, the “Add New Employee” pushbutton is “Ungrayed” and now can be selected (if desired).

3.) User types several other characters (already exists in list):



<u>Note:</u> Although text does appear in the “New Employee” field, it already exists in the “Employees List”. Therefore, the “Add New Employee” pushbutton is “Grayed” and cannot be selected.

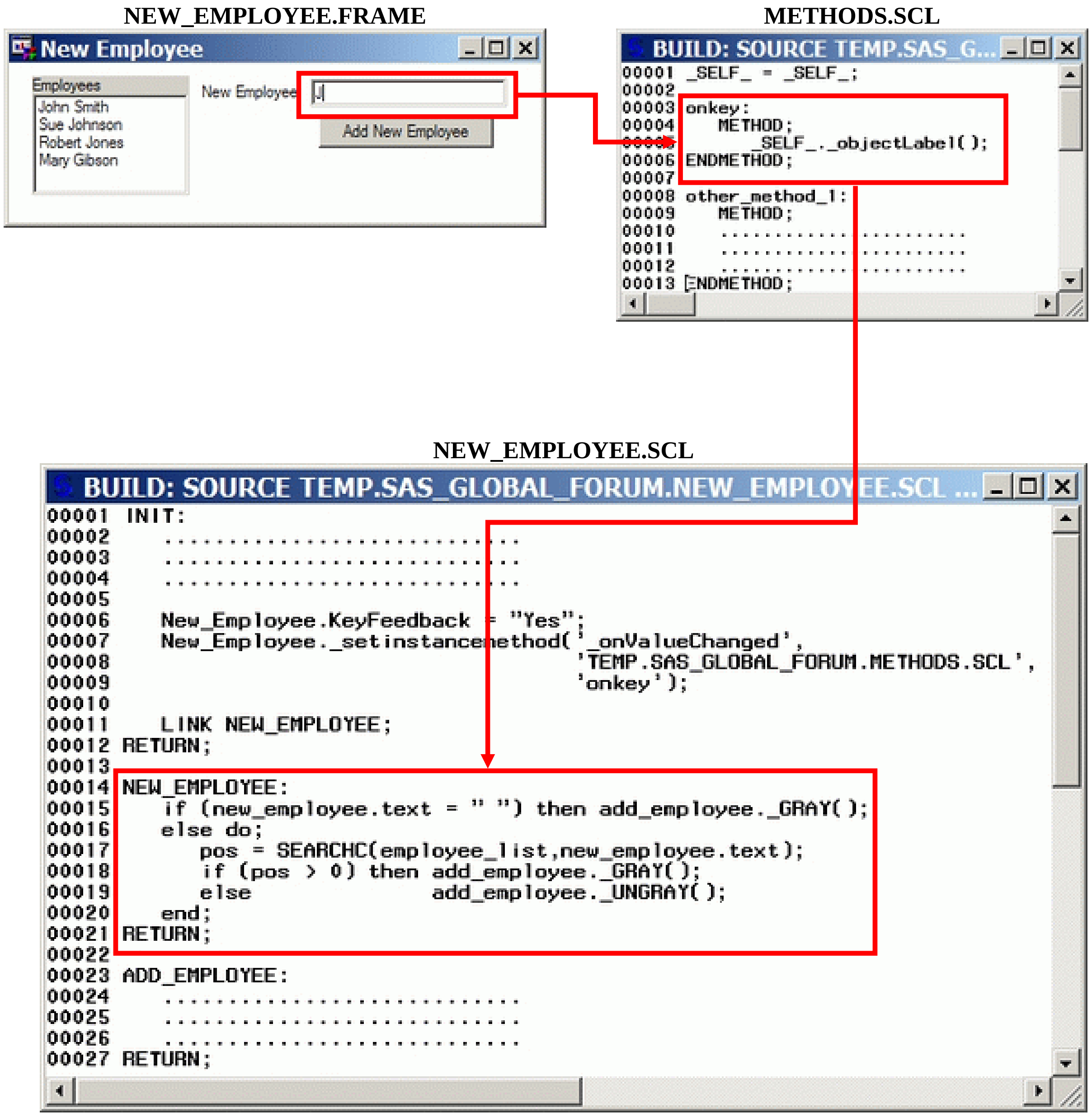
4.) User types several other characters (does not exist in list):



<u>Note:</u> Since the typed text does not appear in the “Employees List”, the “Add New Employee” pushbutton is “UnGrayed” and can now be selected to add the new entry to the Employee List.

SUMMARY

Once **Key Feedback** has been set up for the text entry object, the following occurs. As each keystroke is typed, SCL code is run in the “Methods.SCL” program (“onKey” Method). This code redirects control back to the Frame SCL program; in particular the labelled section for the text entry object, which would then run as programmed. And to stress again, this would occur after each and every keystroke typed in the New_Employee text entry field.



NEW_EMPLOYEE.FRAME

METHODS.SCL

```
_SELF_ = _SELF_;

onkey:
   METHOD;
      _SELF_._objectLabel();
ENDMETHOD;

other_method_1:
   METHOD;
    ............ ..........
    ............ ..........
    ............ ..........
ENDMETHOD;
```

NEW_EMPLOYEE.SCL

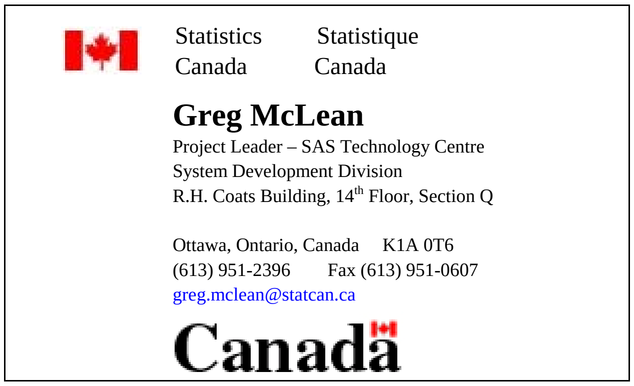
```
INIT:
    ............................
    ............................
    ............................

    New_Employee.KeyFeedback = "Yes";
    New_Employee._setinstancemethod('_onValueChanged',
                                    'TEMP.SAS_GLOBAL_FORUM.METHODS.SCL',
                                    'onkey');

    LINK NEW_EMPLOYEE;
RETURN;

NEW_EMPLOYEE:
    if (new_employee.text = " ") then add_employee._GRAY();
    else do;
       pos = SEARCHC(employee_list,new_employee.text);
       if (pos > 0) then add_employee._GRAY();
       else              add_employee._UNGRAY();
    end;
RETURN;

ADD_EMPLOYEE:
    ............................
    ............................
    ............................
RETURN;
```

Although the example used in this paper is quite trivial, it introduces the concept of **Key Feedback** and opens up an endless list of possible uses when developing SAS/AF applications. Using this technique you will no longer have users saying:

“Do I hit the Return Key or Not?”

CONTACT INFORMATION

For information on topics covered in this paper please contact:



Statistics Canada   Statistique Canada

Greg McLean
Project Leader – SAS Technology Centre
System Development Division
R.H. Coats Building, 14th Floor, Section Q

Ottawa, Ontario, Canada   K1A 0T6
(613) 951-2396   Fax (613) 951-0607
greg.mclean@statcan.ca

Canada