User Interface - Bug #3559

trigger matching for multi-label events (i.e. RETURN/ENTER/CTRM-M, PAGE-UP/PGUP/PREV-PAGE/PREV-SCRN)

05/01/2018 05:36 PM - Constantin Asofiei

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 05/01/2018 05:46 PM - Constantin Asofiei

4GL has a weird way of matching an event to a trigger, if that event has multiple labels associated with it. For example, a APPLY "RETURN" will match to a ENTER trigger, but only if there is no explicit RETURN trigger. There is also a special case for AUTO-RETURN, where it will match **only and only** to a ENTER trigger, and not RETURN trigger.

For example, GuiKeyboard.standardKeyFunctions defines multiple events with more than one function - RETURN, COPY, CUT, PAGE_UP, PAGE_DOWN, PASTE. We need to find the rules of trigger matching based on the defined triggers - I assume we will need to test misc combinations of defined triggers, apply and keyboard-level raised events.

An assumption is that each case has a 'default' trigger which will be matched if an explicit trigger is not found; but I don't understand why an APPLY "PG-UP" will match to a PREV-SCRN trigger even if there is an explicit PG-UP trigger. Maybe there is a specific order in which event labels are searched for triggers? And if no trigger found for a label, go to the next one (and labels are sorted in some pre-determined way)?

In FWD's trigger matching, there is this code in EventList.lookupWorker and EventList.addEvent:

```
         caseEvent = setCase(event[i]);

         // alternate key labels won't be matched if we don't convert them to the primary label

         // convert the label to a key code, alternate labels are honored here
         int code = Keyboard.keyCode(caseEvent);

         // is this a valid label?
         if (code != -1)
         {
            // yes, it was a label so now convert the common code to the primary label name
            String primary = Keyboard.keyLabel(code);
            caseEvent = (primary.length() > 0) ? primary : caseEvent;
         }
```

This converts an event to its 'primary label' - the lookupWorker is a recent change which fixes a APPLY "RETURN", as the trigger was computing the ENTER primary label and APPLY "RETURN" was not matching to it - but the problem is more complex than this.