User Interface - Support #11864

review and correct UI event gap marking

09/10/2026 07:38 PM - Greg Shah

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

Description

History

#1 - 09/10/2026 07:48 PM - Greg Shah

Proposed Gap Marking Support Levels: 4GL UI Event References

Purpose

The **4GL UI Event References** report now carries support level columns. This document records how the levels were derived and, more importantly, lists the named events which are still **deliberately unmarked** so the team can supply their levels.

See Gap Analysis for the meaning of each support level.

The changes are committed in 11747a revision 16741. Please review and correct.

Why this report needed a different mechanism

Every other gap marking map is keyed by AST token type. An EVENT node has no distinguishing token type: the parser accepts any symbol, string or single character and validates it against Keyboard.registeredEventName(). Marking therefore has to be keyed by the event **name**.

The names are also not one homogeneous set. FWD itself splits them two ways, and the split matters for gap analysis:

- **Keyboard keys and key functions** (ALT-A, CTRL-F8, F12, TAB, RETURN, GO ...) resolve through Keyboard.keyCode(). These are handled generically by the keyboard mapping, so they are all supported to the same degree and enumerating them would be busywork that goes stale as the keyboard tables grow.
- **Named events** (CHOOSE, VALUE-CHANGED, OLE-DRAG-DROP ...) resolve through Keyboard.eventCode(). Each is implemented, or not, on its own.

What was changed

- src/com/goldencode/p2j/ui/Keyboard.java - new classifyEventName() returning "key", "event" or "unknown", following the same precedence as EventList.eventCode() (a name which resolves to both is treated as a key). The keyboard initialisation that registeredEventName() was doing inline was extracted into initKeyboards() and is now shared.
- src/com/goldencode/p2j/uast/ProgressPatternWorker.java - new eventKind() library method exposing the classifier to TRPL as prog.eventKind().
- rules/gaps/user_interface.rules - new addUiEvents() function holding the named event levels.
- rules/gaps/gap_analysis_marking.xml - marking rule for EVENT nodes.
- rules/reports/profile.rpt - supportLvlExpr enabled on the report.

Calling Keyboard from conversion is not new: the parser already calls registeredEventName() on every event it matches, so the keyboard initialisation is known to work in that context.

How the levels were assigned

Group	Count	Level	Basis

Keyboard keys and key functions	66	Full / Full	classified at marking time, not enumerated; the keyboard mapping handles them generically and the parser already rejects names no keyboard registers
Named events with implementation evidence	28	Full / Full	each is referenced by the FWD event implementation - EventList, EventDefinition, the widget GuiImpl classes, the browse and editor support, or (for READ-RESPONSE) util/SocketImpl
Named events registered with no runtime	4	Full / None	DESELECT, OFF-END, OFF-HOME, PARENT-WINDOW-CLOSE; the 20210121 history entry in Keyboard.java states plainly that no runtime exists for them
COM/OCX drag and drop events	6	Full / None	the native COM bridge has no connection point sink, so no COM event is ever delivered to converted code
Named events left unmarked	98	Unknown	see below

Named events deliberately left unmarked

I only marked events where there was concrete evidence in the FWD source. An attempt to infer support for the rest by counting references to their event code constants was tried and **abandoned as unreliable**: VALUE-CHANGED, which is unquestionably implemented, has zero literal references because everything goes through the SE_ and KA_ constants. Rather than invent 98 levels from a signal that demonstrably produces false negatives, these are left Unknown so the report shows them as needing attention.

These are the ones needing a level:

ABORT, AFTER-LABEL-EDIT, ANY-KEY, APPEND-LINE, BEFORE-LABEL-EDIT, BLOCK, BOTTOM-COLUMN, BREAK-LINE, CANCEL-EDIT, CANCEL-PICK, CHANGE-NODE-DIRECT, CHANGE-TOP-VISIBLE-NODE, CHOICES, COLUMN-SORTING, COMPILE, CONNECT, DEFAULT-POP-UP, DELETE-COLUMN, DELETE-END-LINE, DELETE-FIELD, DELETE-LINE, DELETE-WORD, DROP-FILE-NOTIFY, EDITOR-BACKTAB, EDITOR-TAB, END-BOX-SELECTION, END-EDIT, END-MOVE, END-ROW-RESIZE, END-SEARCH, ENTER-MENUBAR, ERROR, FIND-NEXT, FIND-PREVIOUS, GET, GOTO, HELP, INSERT-COLUMN, INSERT-FIELD, INSERT-FIELD-DATA, INSERT-FIELD-LABEL, INSERT-MODE, MAIN-MENU, MODIFIED, MOUSE-CLICK, MOUSE-DBLCLICK, MOUSE-DOWN, MOUSE-UP, MOVE, NEW, NEW-LINE, NEXT-ERROR, NEXT-FRAME, NEXT-WORD, NODE-CHECK, NODE-CLICK, NODE-COLLAPSED, NODE-COLLAPSING, NODE-EXPANDED, NODE-EXPANDING, OBJECT-CLICKED, OBJECT-LOST-FOCUS, OBJECT-VALUE-CHANGED, OPEN-LINE-ABOVE, OPTIONS, PAGE-LEFT, PAGE-RIGHT, PEN-DOWN, PEN-UP, PICK, PICK-AREA, PICK-BOTH, PREV-FRAME, PREV-WORD, PROCEDURE-COMPLETE, PUT, RECALL, REPLACE, REPORTS, RESUME-DISPLAY, SAVE-AS, SCROLL-LEFT, SCROLL-MODE, SCROLL-NOTIFY, SCROLL-RIGHT, SELECT-ALL, SELECTION-CHANGED, SETTINGS, START-EDIT, START-MOVE, START-ROW-RESIZE, STOP, STOP-DISPLAY, TOP-COLUMN, TOP-LEFT-CHANGED, UNIX-END, WEB-NOTIFY, WEB-UPLOAD-COMPLETE

Questions

1. Are the 28 named events marked Full / Full acceptable at that level, or do some deserve Basic or Partial? The evidence establishes that FWD implements them, not how completely.
2. Should the 98 unmarked events be worked through in bulk, or left Unknown until a project actually uses one?
3. DESELECT, OFF-END, OFF-HOME and PARENT-WINDOW-CLOSE are marked Full / None on the strength of a 2021 comment. Is that still accurate?

#2 - 09/10/2026 07:48 PM - Greg Shah

We should write testcases that prove that each event is supported at runitme.