User Interface - Feature #12042

OCX replacement infrastructure: registry profiles, .wrx decoding, stored properties and COM runtime gaps

10/20/2026 05:41 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 - 10/20/2026 05:41 PM - Greg Shah

OCX replacement infrastructure: registry profiles, .wrx decoding, stored properties and COM runtime gaps

Project: User Interface

Problem

#11919 implemented the first ActiveX replacement, ctToolBar, together with the mechanism the other replacements build on: the project-level OCX registry (ocx-registry.xml), the decoding of the .wrx design-time state at conversion, and member resolution against the target class. Replacing every control a large application uses needs more from that mechanism than one control did, and some work is common to all the controls rather than part of any one of them.

Work

1. **Finish the #11919 findings:**
   - the decoded .wrx resources (JSON and image files) must reach the run-time classpath of a converted project and be registered as conversion artifacts;
   - the 4GL error raised for an unknown COM member is built in three places and differs from comhandle.showAccessError; use one shared helper;
   - a program with several .wrx files, and the forms of LoadControls(SEARCH(...)) that compute the file name, must resolve.
2. **Registry profiles.** A profile per source control, since several controls share one target widget; lookup by control name as well as CLSID; matching of event handlers to the control they belong to; and **no-op entries** for controls that are being dropped, so that code which still loads them converts and runs.
3. **Generate the CLSID mapping** from the .wrx inventory of a project, rather than writing it by hand.
4. **.wrx decoder gaps:** ICO, GIF and JPEG images, every image slot, and the toolbar item records.
5. **Stored properties of controls that keep no property names.** Some controls persist their design-time properties through the MFC COleControl stream, which carries values only. Read them through the live control on a Windows machine with the control installed, record the mapping of stored slot to property name per control type, and feed it into the decoder.
6. **COM runtime gaps** that several replacements depend on: ENABLE-EVENTS, event name resolution, and value conversions between COM and 4GL types, COM dates in particular.
7. **Behaviour checks against the real controls** in Windows OpenEdge, where several replacements depend on an answer: whether setting a value from code fires events (check-box cascades, tab changes, dates), a blocking context menu inside another control's event handler, tree index behaviour, grid edit order, and timer ticks while a dialog is open. Record each answer for the control task that needs it.
8. **A regression pass** over one or two real screens per use-case pattern once the controls are replaced, in addition to the per-control tests.
9. Update the OCX emulation playbook attached to #11919 with what changes.

Testcases

- A converted program loading two .wrx files and a computed .wrx name, run end to end with its decoded resources.

- A dropped control registered as a no-op: the program converts and runs, and member accesses do nothing.
- The image formats and slots above decoded from real .wrx files and compared with the images the control shows on Windows.
- COM events with ENABLE-EVENTS, and a COM date round-tripped to a 4GL DATE and DATETIME.

#2 - 11/16/2026 04:07 PM - Greg Shah

One more item for the common OCX work. We have a project that opens the 4GL popup menu of an ActiveX control-frame by sending it a right mouse button up message, after the control's own mouse event:

```
RUN SendMessageA (chCtrlFrame:HWND, 517, 0, 0, OUTPUT iResult).   /* WM_RBUTTONUP */
```

The POPUP-MENU is attached to the control-frame, and OpenEdge shows it when the control-frame receives WM_RBUTTONUP. #12040 adds SendMessageA message 517 to the Win32 emulation. With the controls replaced by FWD widgets, the message sent to a replaced control-frame's HWND must still show the control-frame's popup menu, at the current mouse position, exactly as for a native widget. This should work for every replaced control, so it belongs in the common infrastructure rather than in each control's task.

Testcases: a replaced control (for example a list grid and a tree) with a POPUP-MENU on its control-frame, SendMessageA 517 sent from the control's right mouse event, on the web and Swing clients; the menu appears at the mouse position and its choose triggers run.