User Interface - Feature #12056

Microsoft WebBrowser ActiveX control: script execution and document access

10/20/2026 06:13 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 06:13 PM - Greg Shah

Microsoft WebBrowser ActiveX control: script execution and document access

Project: User Interface

Problem

The Microsoft WebBrowser control (Shell.Explorer, CLSID {8856F961-340A-11D0-A96B-00C04FD705A2}) is used to embed maps, a hosted login page and chart panes. Members used: Navigate, Document:parentWindow:execScript (two-way JavaScript calls), Document:Script:<name> to read values back from the hosted page, ExecWB to print, and position and size. Events: TitleChange and DocumentComplete.

FWD's HTML-BROWSER renders a live iframe with navigation, loading from a string, printing, a postMessage bridge and navigation events. It has no execScript and no access to the page from the server.

Work

1. Map the control onto HTML-BROWSER with an alias table.
2. Add a script execution bridge for execScript and for reading script values from the page, over the postMessage bridge, with a completion event so that the 4GL can wait for the result.
3. Map ExecWB printing, TitleChange and DocumentComplete.
4. Register the control's CLSID in the OCX registry (ocx-registry.xml), mapping it to the target widget, as #11919 did for ctToolBar.
5. Decode the design-time state the control keeps in the .wrx storage at conversion time, so that it exists after conversion.
6. Implement the members and events listed above on the target, with the vendor's semantics: argument types, return values, defaults, and the order in which events fire. Members of the type library that are not used become no-ops with a conversion warning, as the OCX pipeline already does.
7. Follow the OCX emulation playbook attached to #11919: an OpenEdge test program with goldens for each member and event, recorded input sequences for the interactive parts, and Lyra automation.
8. Update the gap marking and the OCX registry documentation.

Testcases

Each member and event above, exercised on Windows OpenEdge with the real control to produce goldens, then run against the FWD replacement on the web and Swing clients. The vendor help files are the reference where behaviour cannot be observed directly.