User Interface - Feature #12047

replacement for the DBI ctTree ActiveX control (tree with columns)

10/20/2026 05:49 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 05:50 PM - Greg Shah

Replacement for the DBI ctTree ActiveX control (tree with columns)

Project: User Interface

Problem

ctTree (ctTree.ocx 5.0) is a hierarchical picker with columns, populated at run time, typically by recursive loops over temp-table queries. Members used include AddNode, NodeCargo and NodeData carrying keys, NodeIsParent, NodeExpanded, NodeParent, NodePicture and AddPictureNode, CellCheck with cascading CheckClick, CellText, NodeFontBold, ClearNodes, EnsureVisible, RemoveNode and NodeShowChecks — about 86 distinct members. The column headers are set at design time in the .wrx (CColumnHeader records). Events: CheckClick, RightClick, NodeClick, DblClick, Change, KeyDown, Expand. A common composite is a tree that repopulates a ctList on NodeClick.

FWD's TREELIST has columns, check boxes, icons, multi-select, drag and drop and a right mouse down event.

Work

1. Implement a new ctTree widget class on FWD's tree base (the TREEVIEW / TREELIST infrastructure), exposing the DBI node API directly rather than through an alias table: nodes addressed by index, cargo and data, columns, check cells with cascading, per-node pictures and bold font (see #4730).
2. Take the columns from the .wrx.
3. Register the control's CLSID in the OCX registry (ocx-registry.xml), mapping it to the target widget, as #11919 did for ctToolBar.
4. Decode the design-time state the control keeps in the .wrx storage at conversion time, so that it exists after conversion.
5. 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.
6. 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.
7. 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.