# User Interface - Feature #12043

## replacement for the DBI ctList ActiveX control (list grid)

10/20/2026 05:43 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 05:43 PM - Greg Shah**

# Replacement for the DBI ctList ActiveX control (list grid)

**Project:** User Interface

## Problem

ctList (DBI Technologies Component Toolbox, ctList.ocx 5.0) is a multi-column list grid populated at run time. Applications that use it heavily rely on:

- **Rows:** AddItem with caret-delimited cell text, ListCargo carrying a key per row, Clearlist, SELECTED, listindex, listcount.
- **Cells:** CellText, CellCheck for check cells, CellPicture for per-cell icons.
- **Columns:** defined at design time in the .wrx (CColumnHeader records) in most screens, and at run time with AddColumn in a few. Header-click sorting through SortColumn, SortDirection, SortList and ColumnSortable.
- **Editing:** in-place cell editing in a few screens.
- **Events:** CheckClick, ItemClick, RightClick, ItemDblClick, Change, KeyDown. RightClick usually opens a ctDropMenu.
- Cell icons come from the embedded bitmaps in the .wrx or from a ctImage.

FWD has the TREELIST infrastructure and a LISTVIEW widget built on it (#3857), neither of which has check cells, per-cell pictures, per-row cargo, header sort state, a right-click event or in-place editing.

## Work

1. Build a list grid widget on the TreeList infrastructure covering the surface above, so that the Microsoft ListView (#3857) can share it.
2. Add check cells, per-cell pictures, per-row cargo, header sort state and sorting, a right-click event, and in-place cell editing.
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.