# User Interface - Feature #12064

## replacement for the DBI ctSched ActiveX control (scheduler)

10/20/2026 06:31 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 06:31 PM - Greg Shah**

# Replacement for the DBI ctSched ActiveX control (scheduler)

**Project:** User Interface

## Problem

ctSched is a Gantt-style time bar scheduler (ctSched.ocx 3.0), used in one service screen with about 29 distinct members.

It implements a scheduling feature in service screens: resources or dates on one axis, time on the other, with bars or appointments that the user selects, double-clicks, moves and resizes. Neither FWD nor the 4GL has a scheduler widget.

## Work

1. Inventory every member and event used, from the converted source, and their semantics from the vendor help files.
2. Implement a scheduler widget for the web and Swing clients covering that surface, including drag and drop of bars or appointments with the vendor's pre-drop veto events. Consider basing it on an established JavaScript scheduling component for the web client.
3. Share the implementation between ctSchedule, ctSched and ctMDay where their models overlap.
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.