User Interface - Feature #12044

replacement for the DBI ctDropDate and ctDEdit ActiveX controls (date and time fields)

10/20/2026 05:44 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 05:44 PM - Greg Shah

Replacement for the DBI ctDropDate and ctDEdit ActiveX controls (date and time fields)

Project: User Interface

Problem

ctDropDate (ctDropDate.ocx 3.0) is a date field with a drop-down calendar. Applications read and write DATE as a days-since-1900 serial, typically round-tripped as DATE("01/01/1900") + h:DATE - 2, plus ENABLED and VISIBLE. Events: Change and KeyPress.

ctDEdit (ctDEdit.ocx 5.0) is a masked time-of-day editor. Applications read and write TIME in seconds since midnight, with -1 meaning unset, plus ENABLED and TEXT. Events: ClickSpin (typically plus or minus 60 seconds per click) and KeyPress (Tab and Enter advance focus). Format and mask are set at design time in the .wrx only.

The two are often paired, for a date plus time, or two ctDropDate controls for a range. FWD's CALENDAR widget models the Microsoft date time picker, with drop-down and up-down styles and a time format, and already emits Change and KeyPress.

Work

1. Map both controls onto CALENDAR.
2. Add a serial-date accessor for DATE, so that the 4GL date arithmetic above keeps working, and a seconds accessor for TIME with the -1 sentinel.
3. Add the ClickSpin event, and take the format and mask from the .wrx.
4. Check focus advance on Tab and Enter.
5. Register the control's CLSID in the OCX registry (ocx-registry.xml), mapping it to the target widget, as #11919 did for ctToolBar.
6. Decode the design-time state the control keeps in the .wrx storage at conversion time, so that it exists after conversion.
7. 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.
8. 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.
9. 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.