User Interface - Feature #12040

WIN32 API emulation: additional user32, gdi32 and wtsapi32 entry points

10/20/2026 02:32 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 02:33 PM - Greg Shah

WIN32 API emulation: additional user32, gdi32 and wtsapi32 entry points

Project: User Interface

Problem

FWD emulates a set of Win32 entry points that GUI applications call through PROCEDURE ... EXTERNAL, in util/NativeAPIEmulation.java: among them LockWindowUpdate, PostMessageA, ShowScrollBar, GetWindowRect, GetParent, ScreenToClient, GetCursorPos, GetSystemMetrics, DisableProcessWindowsGhosting, GetComputerNameA and the wsock32 host name calls. The handle they receive is the FWD widget id that HWND returns.

Several entry points that real applications use are missing or only partly supported. A call to one of them reaches UnimplementedFeature or does nothing.

Work

Add the following to NativeAPIEmulation, each acting on the client where the user is:

1. **SendMessageA**, which today handles only WM_HSCROLL, WM_VSCROLL and 516/517:
   - WM_LBUTTONDOWN / WM_LBUTTONUP (513/514), through the existing applyMessageA path, as PostMessageA already does;
   - CB_SHOWDROPDOWN (335), which opens or closes a combo box drop-down.
2. **GetDC, ReleaseDC and gdi32 GetDeviceCaps**: return the client's logical DPI for LOGPIXELSX / LOGPIXELSY (88/90). Applications use this to scale mouse coordinates for ActiveX hit testing.
3. **GetSysColor**, which today handles indices 15, 16 and 20: add COLOR_MENU (4) and the other common indices, from the theme palette.
4. **Borderless window**: GetWindowLongA / SetWindowLongA on GWL_STYLE, AdjustWindowRect, and SetWindowPos with SWP_FRAMECHANGED, mapped onto window attributes: no title bar, thin border, and the requested position. Applications typically combine the style bits through helper DLL functions (for example bit removal and bitwise OR in a "proextra" library loaded with LoadLibraryA), so those calls need handling too.
5. **System menu**: GetSystemMenu, GetMenuItemCount, RemoveMenu and DrawMenuBar, so that an application can disable a window's close button.
6. **WTSQuerySessionInformationA** for classes 10, 14 and 26 (client name, client address, session information), answered from the FWD client's identity.
7. **Inactivity**: GetLastInputInfo from the client's input timestamps, and WaitForInputIdle as a no-op.

Each entry point needs testcases that compare the result with Windows OpenEdge.

#2 - 11/16/2026 04:07 PM - Greg Shah

One more case for the wsock32 entry points. We have a project with 4GL SMTP code that runs as a batch or appserver process and calls gethostname (after WSAStartup) to get the local host name for the SMTP HELO / EHLO command.

The emulation answers gethostname, gethostbyname and inet_ntoa for the **client** machine, which is right for interactive code that wants to identify the user's PC. For server-side code the SMTP server needs the name of the machine that actually connects to it, which is the FWD server.

Please define which side answers these calls in each kind of session: the client machine for interactive sessions, and the server for batch processes and appserver agents, which have no user's PC behind them. If a project needs the other answer in a given case, provide a way to get it (for example a configuration setting, or a documented alternative such as OS-GETENV("HOSTNAME") on the server).

Testcases: gethostname / gethostbyname / inet_ntoa in an interactive web client session, a batch client, and an appserver agent, checking which host each returns.