User Interface - Feature #12041

emulate ShellExecuteA: deliver documents to the user's browser

10/20/2026 03:04 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 03:04 PM - Greg Shah

Emulate ShellExecuteA: deliver documents to the user's browser

Project: User Interface

Problem

GUI applications commonly call the Win32 ShellExecuteA entry point from shell32.dll to open a document with its registered application, or to print it:

```
PROCEDURE ShellExecuteA EXTERNAL "shell32.dll":
   DEFINE INPUT  PARAMETER hwnd         AS LONG.
   DEFINE INPUT  PARAMETER lpOperation  AS CHARACTER.
   DEFINE INPUT  PARAMETER lpFile       AS CHARACTER.
   DEFINE INPUT  PARAMETER lpParameters AS CHARACTER.
   DEFINE INPUT  PARAMETER lpDirectory  AS CHARACTER.
   DEFINE INPUT  PARAMETER nShowCmd     AS LONG.
   DEFINE RETURN PARAMETER hInstance    AS LONG.
END PROCEDURE.

RUN ShellExecuteA (0, "open", cPdfFile, "", "", 1, OUTPUT iResult).
```

NativeAPIEmulation has no shell32 entry points. In a converted system the file is usually written on the server, where the 4GL logic runs, and the user is at a web browser with no access to the server's file system. The natural equivalent is to deliver the file to the browser. FWD already has the mechanism: openMimeResource, used by OPEN-MIME-RESOURCE and by WebBrowserManager.openURL.

Work

1. Add ShellExecuteA to NativeAPIEmulation.
2. For the open operation (and a null or empty operation, which means the default verb), deliver the named file to the user's browser through openMimeResource, either as a download or opened in a new tab, chosen by its MIME type. A URL (http:, https:, mailto:) is opened in the browser.
3. For the print operation, deliver the file the same way, so that the user prints it from the browser.
4. Return a value above 32 on success and the documented error codes on failure, for example a file that does not exist.
5. Decide what to do for an executable or a batch file, which cannot be run on the user's machine from the server: return an error code, and log the call.
6. Check the Swing GUI client, where the file can be opened with the desktop's registered application.

Testcases

- open and print for a PDF, a spreadsheet, a text file and an image, a URL, a missing file, and an executable, on the web and Swing clients, checking the return value against Windows OpenEdge.

#2 - 11/15/2026 03:48 PM - Greg Shah

The same need also reaches FWD through .NET instead of shell32.dll. A project prints a document with System.Diagnostics.Process and the shell verb:

```
oProcess = NEW System.Diagnostics.Process().
oProcess:StartInfo:FileName        = cDocument.
oProcess:StartInfo:UseShellExecute = TRUE.
oProcess:StartInfo:Verb            = "print".
oProcess:Start().
```

FileName is a document, not a program, and Verb is the Windows shell verb, so this is ShellExecute under another name. It should share the implementation in this task:

1. Provide replacement classes for System.Diagnostics.Process (Process(), StartInfo, Start(), EnableRaisingEvents), ProcessStartInfo (FileName, Verb, UseShellExecute, CreateNoWindow, WindowStyle) and ProcessWindowStyle, and map them in the conversion.
2. When UseShellExecute is true and FileName is a document, Start() with the verb open, print or none delivers the file to the user's browser exactly as ShellExecuteA does here.
3. When FileName is an executable, follow the same rule as ShellExecuteA for executables: fail, and log the call. CreateNoWindow and WindowStyle have no effect.

Testcases: the ShellExecuteA cases above, repeated through Process with each verb.