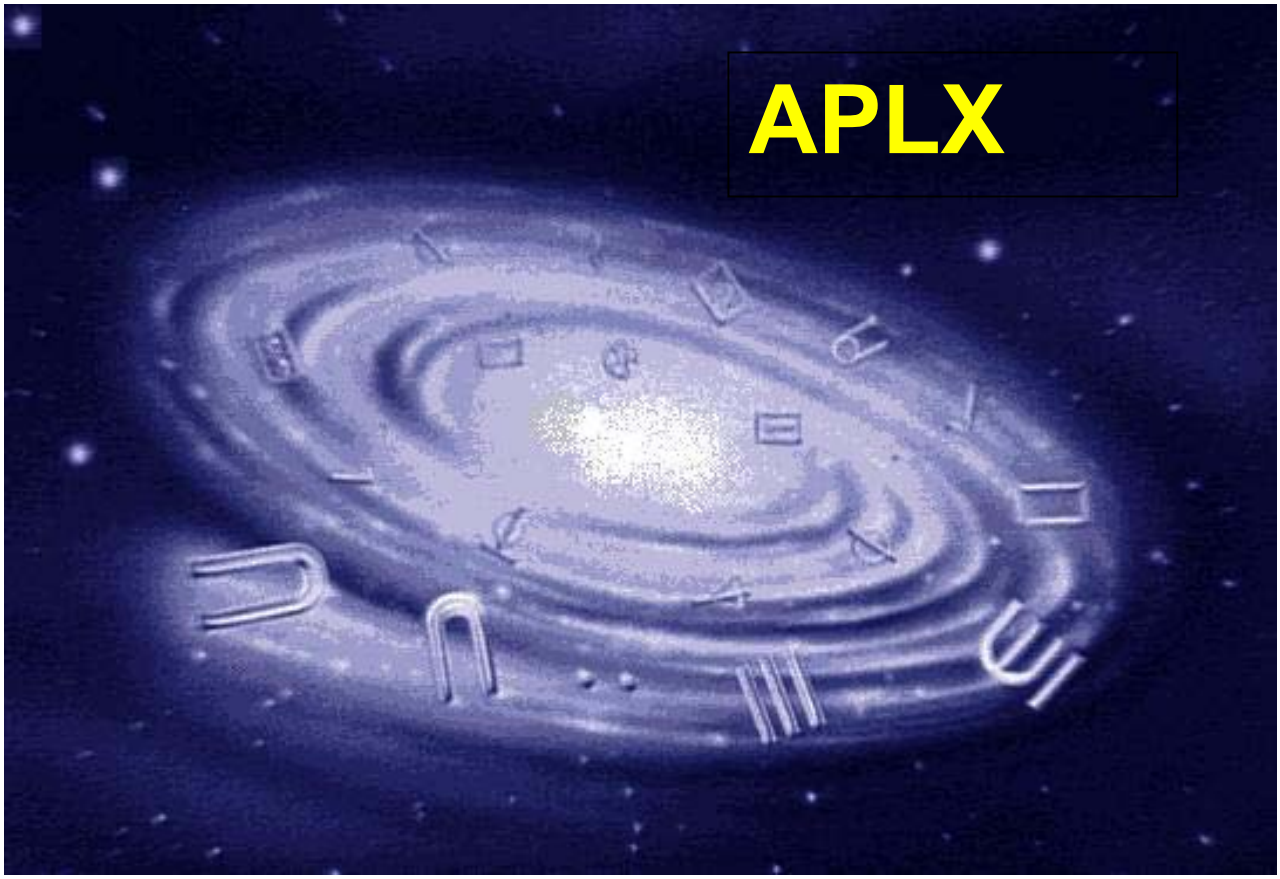


APLX Standalone Applications

Version 4.0

MICRO APL

MicroAPL welcomes your comments and suggestions.
Please visit our website: http://www.microapl.co.uk/apl

Version 4.0 November 2007

Contents

Building Standalone APL Applications

What is a standalone APL application?

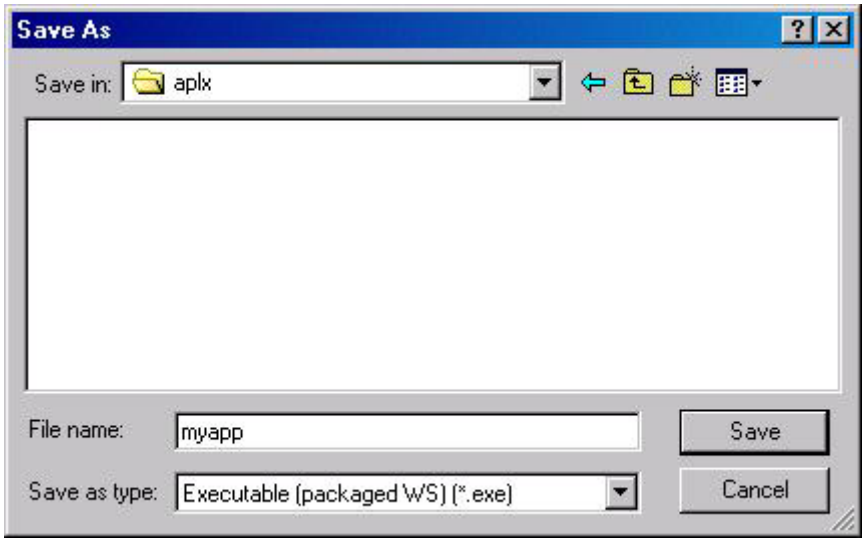
If you write an APL application using APLX, you can build a standalone application from it using a facility called the *APLX Packager*, which is built-in to APLX. This takes an APL workspace, and packages it together with a special runtime version of the APLX interpreter to form a single file which is a program which runs under Windows, Linux or MacOS. You can use this facility to create applications where the users need not be aware that the application was written in APL, and, subject to some simple conditions described in the section Redistribution of Standalone Applications, you can distribute the application to other people without having to pay a royalty to MicroAPL.

Note: The APLX Packager is not available in the free Evaluation or Personal Edition versions of APLX. You need to purchase the appropriate full version of APLX to create standalone applications.

Creating the standalone executable file

To create a standalone application, you should first develop and test your APL workspace as normal on the target platform (Windows, MacOS or Linux), ensuring that you have set `⎕LX` so that it starts automatically when it is loaded.

Then you simply select the 'Save As..' option from the APLX File menu, and use the drop-down menu in the dialog to select the save format to be "Executable (packaged WS)":



You can use any name which the operating system permits under MacOS or Linux; under Windows, the name must end in `'.exe'` (this will be added automatically).

An example of a simple standalone application and the workspace from which it was created is supplied with APLX. The workspace is `10 PACKAGEDEMO` and the standalone application is

`'SampleApp.exe'` (under Windows) or `'SampleApp'` under MacOS and Linux. The packaged application can be found in the `/bin` directory of the APLX installation.

Support libraries

Windows

Under Windows, the end-user needs two DLLs (Dynamic Link Libraries) to run the application. The library `aplxrun4.dll` is supplied with your copy of APLX, and can be distributed with your application code. A second library called `APLXSupport4.dll` is also needed; this is available free of charge (subject to a licence agreement) from the MicroAPL website http://www.microapl.co.uk/apl/player. Both DLLs should be installed in one of the following locations:

- In the same directory as your standalone application;
- In the standard Windows DLL path;
- In the `/bin` directory of the standard APLX installation.

So that you can test your application, the DLLs are installed as part of the full version of APLX.

Linux

Under Linux, the end-user needs a file called `APLXSupport4` in order to run the application. Again this is available free of charge (subject a licence agreement) from the MicroAPL website http://www.microapl.co.uk/apl/player. It should be installed in one of the following locations:

- In the same directory as your standalone application;
- In the standard Linux executable path;
- In the `/bin` directory of the standard APLX installation.

So that you can test your application, APLXSupport4 is installed as part of the full Linux version of APLX.

MacOS

Under MacOS, the application you build in this way is completely self-contained; all of the APLX files and libraries required to run it are included in the application file.

Workspace size

Before saving the workspace as an executable, it is a good idea to set the preferred workspace size which will be used when the application runs. You can do this using `7 ⎕CONF`.

File types and icons

MacOS

Under MacOS, your application will by default have a type `'APPL'` but a blank signature. This means that it will not respond to document files being dragged on to it, or double clicked. If you want to handle events of this kind, you will need to use Apple's ResEdit (or a similar tool) to set up the `BNDL` resource for your packaged application, and set the file signature to a unique four-character code. You can also use ResEdit to define a custom icon for your program.

Windows

Under Windows, your application will by default have the standard Windows icon for an executable file, and will have no file types associated with it. If you create a Shortcut to the application (by right-clicking on the program name), you can select a different icon for the shortcut. You can associate one or more document file types with the application by double-clicking on a sample file and selecting your application as the one to associate with the file type.

How APLX Behaves in a Standalone Application

Restrictions of the Runtime Interpreter

The APL interpreter which is packaged together with your workspace to form the standalone application is an almost complete implementation of APLX, but without the capability of running in desk-calculator mode and without the APL development environment (debugger, function editor, etc). Thus, your workspace must include a latent expression which takes over when the application starts up. If your APL code ever tries to return to desk calculator mode (either because the latent expression terminates normally, or if an untrapped APL error occurs or breakpoint is hit), then the APL session will end. If it terminates because of an error, the error message will be displayed in a message box.

Other differences are:

- The APL development windows are not available. These include the Debug, Watch, Display, Workspace Explorer, and Control Browser windows.
- Edit Windows can be invoked using `⎕EDIT`, but can only be used to edit variables, not functions.
- `)SAVE` `)IN` and `)OUT` are not implemented.

The Session window

The APLX Session window *is* available in standalone applications, but it behaves in a special way. On start-up, the Session window does not appear, and the normal APLX sign-on message is not displayed. It will remain hidden unless your packaged application performs ordinary APL output to the Session window, or requests input using `⎕` or `⍞` input. It behaves in a similar way to the Session window in the full version of APLX, but with a simplified set of menus.

This provides flexibility for various requirements, for example:

- A simple traditional APL application which uses the Session window for output and `⎕` for input can be packaged into a Standalone application and will work as it does when running under the full version of APLX.
- An APLX server or file-processing application can use the Session window as a 'log' window, outputting messages to it to indicate progress, or can run in the background without any windows.
- A full windowing application, built using System Classes, can run without the Session window ever appearing.

The APLX font

Use of the APLX font in your applications is optional. Applications which use System Classes for user-interface programming probably will not need the APL font, but if you use the Session window for output we recommend that you do install the APLX font on machines running the application.

Using System Classes in a Standalone Application

User-defined windows and dialogs

APL applications which you write will typically use System Classes to create and manipulate windows, dialogs, and graphics. These will work in the standalone application in the same way as they do in the normal APL development environment, except that you should be careful not to rely on the APLX font being installed on the target system unless you specifically ensure that it is installed with your application.

MacOS About, Quit and Preferences menu items

Under Mac OS X, the 'About, 'Preferences' and 'Quit' menu items reside in the Application menu. (They are created automatically by the operating system). Because these menu items are not created under the control of your APL code, you cannot associate an APL callback function with them in the same way as you would for your own menus. APLX therefore provides three callbacks in the System object, which are triggered if any of these menu items are selected. They are `onAboutMenu`, `onPreferencesMenu`, and `onQuitMenu`. You should use these callbacks to provide the appropriate functionality in your application, so that your program will respond correctly if the user selects one of these menu items.

Under Windows or Linux, these callbacks can be defined, but will never be invoked.

Determining start-up parameters and responding to Apple Events

If you have packaged your APL workspace as a standalone application, it may be launched by the user dragging one or more documents on to the application icon or double-clicking a document (for this to work, you need to have set up the association between the file type/extension and the application). Under Windows or Linux, the user may also have started the application from the command-line. In either case, the System object's `file` property allows your APL program to respond correctly to these user actions. It is a read-only property, which you should retrieve at the beginning of your APL application code.

Under MacOS, `file` contains the full names of the document or documents which you should open or print. If there are several file names, they will be delimited by carriage-return characters. The `action` property indicates whether your should open or print them. As well as checking these properties when your application starts, you should also respond to any `onOpen` event for the System object. This will be triggered if, once the application is running, the user drags further file icons of the right type on to the application icon (or double-clicks on a file icon). This event corresponds to the 'Open Document' and 'Print Document' Apple Events; the `file` and `action` properties will be updated to indicate the new files which need to be opened or printed..

Under Windows, there are two possibilities. If the user dragged a document icon on to your application icon (or double-clicked the document icon), `file` will contain the full filename of the

document. If the user started your application from the command-line, `file` will contain the command-line options which were entered.

Redistribution of Standalone Applications

The APLX Standalone Application licence agreement

Redistribution of the APLX code contained within an APLX Standalone application is subject to the licence agreement supplied with your APLX product. You should study the agreement carefully before distributing your APLX Standalone application to third parties. Some of the main provisions of the agreement include the following:

- In order to distribute a packaged APLX application, you must have bought and paid for a properly licensed copy of APLX for the platform for which you are distributing the application, and you must have returned the registration card to MicroAPL (or registered on-line). Thus, to distribute MacOS applications, you must have bought and registered a copy of *APLX for MacOS*, to distribute a Windows application you must have bought and registered a copy of *APLX for Windows*, and to distribute a Linux application you must have bought and registered a copy of *APLX for Linux*. If you have an Educational licence for APLX, you cannot make and distribute commercial applications with it, but you can distribute applications for educational and scientific use.
- The workspace must be a genuine application, and not a software development product or simply a wrapper for the APL interpreter. This does not mean that your code cannot use the execute symbol to evaluate user input, but this must be incidental to the application.
- Your documentation must include a copyright notice stating that 'Portions of this software are Copyright (C) 1991-2007 MicroAPL Ltd and its licensors'.
- You must comply with the licence terms for the PCRE (Perl Compatible Regular Expressions) code which is incorporated into APLX. These terms are contained in the file `licence.txt` included with your APLX distribution.
- For the Windows and Linux versions of APLX, you must also comply with the licence terms for the Indy networking libraries. These terms are contained in the file `licence.txt` included with your APLX distribution.

Please note that the above is only a brief summary for general information only; you should refer to the formal Licence Agreement for the specific terms of the licence.

The APLX support libraries for Windows and Linux

Under Windows and Linux, some of the APLX runtime code is contained in the file called `APLXSupport4.dll` or `APLXSupport4`. For legal reasons, you are not allowed to distribute these files to third parties (i.e. to individuals or companies outside your own organization). However, they are available for free download from the MicroAPL website. If a potential user tries to run your application but does not have the appropriate file installed, a dialog box will be displayed indicating how to download it.

Under Windows, the library `aplxrun4.dll` is also needed. You may if you wish distribute this with your application; it is also installed as part of the APLXSupport4.dll installation.

The APLX fonts

The APLX fonts supplied with APLX may be freely distributed with your application.