64-bit Guidelines

Because 64-bit support involves new frameworks with a new, optimizing compiler, 64-bit is considered Beta for its initial release. Please read the known issues at the end of this page. For most projects, 64-bit builds will work just fine and we encourage you to try 64-bit builds for your projects. Should you find issues, be sure to report them to use using Feedback.

Starting with 2015 Release 3, you can now create 64-bit app for OS X, Windows and Linux. In most cases, you'll be able to build a 64-bit version of your existing projects without having to make any changes.

With 64-bit, you apps gain access to much more memory than before (32-bit apps had a 3-4GB limit) and math-based code can see significant speed increases due to the new optimizing compiler. In addition, 64-bit operating systems can run 64-bit apps more efficiently than 32-bit apps since they do not have to load an extra compatibility layer.

Building a 64-bit App

The Inspector for each build target (Windows, OS X and Linux) has a new property called Architecture where you can select the type of architecture you want to build. By default it is "x86 32-bit". To create a 64-bit app, check the box next to the target OS, choose "x86 64-bit" for the Architecture and then press the Build button.

You cannot Run (Debug mode) when the Architecture is set to "x86 64-bit". In order to debug, change the Architecture to "x86 32-bit".

64-bit builds using the LLVM compiler which can give you significant performance improvements for math-based code.

You may notice that it takes longer to build a 64-bit app than it does to build a 32-bit app. Because the new compiler is optimizing your app build, the compilation speed will be somewhat slower than the non-optimizing 32-bit compiler. At no point is the compiler "hung" or "stuck", so just wait for it to finish. Over time expect the 64-bit build times to be reduced as the compiler gets fine-tuned.

To make better use of hardware resources, the LLVM compiler uses multiple CPU cores when compiling.

Considerations

Compiler Constants

In a 32-bit app, Target32Bit returns True and Target64Bit returns False.

In a 64-bit app, Target64Bit returns True and Target32Bit returns False.

Use these two compiler constants to determine the type of app.

Integers

in a 64-bit app, the Integer data type is an alias for Int64. In a 32-bit app, the Integer data type is an alias for Int32.

The type of app you need to create varies based on the operating systems you need to support.

OS X

All versions of OS X supported by Xojo (10.7 or later) are already 64-bit. If your app is working properly as a 64-bit app, then you have no reason to continue creating 32-bit apps. Simply create a 64-bit app and make that available to your users.

Windows

Windows is available in both 32-bit and 64-bit versions. The 32-bit version of Windows cannot run 64-bit apps, but the 64-bit version of Windows can run both 32-bit and 64-bit apps.

This means you will likely need to create both 32-bit and 64-bit versions of your Windows apps so that your users can choose the appropriate one for their version of Windows.

Keep in mind that the TargetWin32 compiler constant returns True on Windows, regardless of whether the app is 32-bit or 64-bit. This constant is indicating that the Win32 system is being used and not an indicator of the type of system. To check if the app is 64-bit, use the Target64bit compiler constant.

Build Files

For 64-bit builds, the compiler places framework DLLs next to the executable and not into the "MyApplication Libs" (or just "Libs") folder like it did for 32-bit builds.

A little history: For 32-bit Windows builds, the compiler doesn't actually spit out a native PE32 file. What it does is writes out a stub executable and then tacks all of the program data onto the end of it. The stub is responsible for loading that data into memory and performing all of the relocations and import binding that the OS loader would normally do. The stub doesn't have to link directly against any of the libraries referenced by the program (including the framework), so Xojo has the freedom to put the DLLs required by the program anywhere. However, this approach occasionaly gets apps flagged as malware by antivirus programs and prevents some features Windows users have wanted for ages (like the ability to edit the app manifest).

For 64-bit Windows executables, the compiler is using a native linker to generate actual PE32+ files. These executables link directly against whatever DLLs it needs and are therefore bound to the operating system's search path. In order for the loader to find the DLLs the executable need to launch, they have to be in the same folder as the executable itself.

Attempting to the make the Windows build structure look more like OS X's bundles is unlikely to happen, primarily due to the DLL issue but also because in Windows, the directory is the application bundle . Microsoft expects programs to use installers and doesn't really care how 'messy' your program's folder is.

That said, the change in directory structure does not impose a requirement that installers are used or that it has to

be on the system disk. The only change is that there will be more stuff in the folder with the executable.

Linux

Linux distributions are also available in both 32-bit and 64-bit versions, although 64-bit versions are far more common. In fact, some Linux distributions are no longer making 32-bit versions available.

32-bit versions of Linux cannot run 64-bit apps. 64-bit versions of Linux can run 32-bit versions of app, but only if the appropriate 32-bit compatibility libraries have been installed. These libraries are not typically installed by default and are not always easy to install.

This means you will likely want to create both 32-bit and 64-bit versions of your Linux apps so that your users can choose the appriate one for their version of Linux.

Declares

In general, you'll want to review any Declares that are used in your project. In particular, if a Declare is using a specific Integer type, such as Int32, it may be possible that the library requires you to use Int64 in a 64-bit app. To simplify this, use the Integer data type which is an alias to Int32 in 32-bit apps and Int64 in 64-bit apps.

On OS X, some Cocoa declare require a floating-point number. The 32-bit libraries use Single, but the 64-bit libraries use Double. There is no data type that functions as an alias for these, so you will have to create two separate Declare statements in these situations, separated with compiler constants.

A 64-bit app can only use 64-bit libraries. Declaring to a 32-bit library from a 64-bit app will fail. Similarly, a 32-bit app cannot use 64-bit libraries.

Plugins

A 64-bit app requires plugins that support 64-bit. If your app uses plugins, you should check with your plugin vendor to see if they have updated versions that support 64-bit.

IDE Script for Building

To make it easier to build all the different versions of your app in one step, you can use the following IDE Script:

```
// Build all target platforms
Const kOSX32 = 7
Const kOSX64 = 16
Const kWin32 = 3
Const kWin64 = 19
Const kLin32 = 4
Const kLin64 = 17
Const kLinARM = 18

Dim path As String
path = BuildApp(kOSX32)
path = BuildApp(kOSX64)
```

```
path = BuildApp(kWin32)
path = BuildApp(kWin64)
path = BuildApp(kLin32)
path = BuildApp(kLin64)
path = BuildApp(kLinARM)

Dim result As String
result = ShowDialog("Build All", _
  "Finished building.", _
  "OK", "", "", -1)
```

This script creates Windows, OS X and Linux builds for both 32-bit and 64-bit plus a Raspberry Pi build (ARM 32-bit) all in one step. You can comment out the parts you do not not need to build.

Current Known Issues

These limitations currently exist with 64-bit apps. They will be resolved in future releases.

- You cannot run a 64-bit app in Debug Mode from the IDE. This means you cannot Run as a 64-bit app. To create a 64-bit app, you must Build.
- 64-bit builds can result in larger app sizes.
- For some projects, 64-bit builds can take a long time to finish. Be sure to let Xojo finish building.
  - To reduce build times, ensure you do not have really long methods that can be refactored and simplified. Project items (classes, Windows, etc.) with large amounts of code can also increase build times. Break them into separate classes to reduce build times.
- Windows 64-bit apps do not have version and icon information.
- XojoScript is not yet supported for any 64-bit app.
- You cannot build 64-bit OS X apps from Windows or Linux IDEs.
- The Tooltips class and tooltips on ListBox do not appear on 64-bit OS X apps.
- Long Split/Join operations on String make have issues. If you do not need multibyte encodings, you can use SplitB. Otherwise, convert the String to Text and then use the Text methods to Split or Join.