If you say that your minimum requirements are the Universal contract, then you need to probe for anything beyond that

devblogs.microsoft.com/oldnewthing/20200326-00

March 26, 2020

Raymond Chen

A few days ago, we saw how to declare your app's minimum requirements in its manifest, so that you won't be deployed to a system that doesn't meet your requirements, as well as how to defer this check until runtime.

The manifest mechanism is a little bit more complicated if you are dealing with contracts outside the Universal contract, though.

```
<TargetDeviceFamily Name="Windows.Universal"
    MinVersion="10.0.10240.0" MaxVersionTested="10.0.18362.0" />
```

If you declare your target device family as `Windows.Universal`, then you're saying that your program can run on anything that supports the Universal platform, without any extension contracts. If your program uses any contracts outside the Universal contract, you'll need to probe for them at runtime.

If you declare your target device family as `Windows.Desktop` or `Windows.Phone` or `Windows.Xbox`, then you're saying that your program runs on that specific device family. You can take advantage of any contracts that were part of that device family's extension SDK as of the minimum version you declare.

One mistake is declaring your app as supporting all Universal devices, even though you rely on classes that are in other contracts, typically contracts that are available only on Desktop, and you forgot to add contract checks to your program to validate that the types exist before you try to use them.

To be fair, this is an understandable error, seeing as Desktop is the only platform that developers are going to have easy access to. The Phone device family died out several years ago, testing on an Xbox is not easy, and Windows 10X is not out yet.

Unfortunately, the result is a bunch of application compatibility bugs against Windows, because an app gets installed from the Store to a non-Desktop device, and the program crashes due to a reliance on a Desktop-only class. It gets counted as an application compatibility bug because “The app worked fine on Desktop, but doesn’t work on non-Desktop systems, so there must be some compatibility issue with those other devices.”

And then the platform team digs in and discovers that the program simply forgot that they were using types outside the Universal contract in an ap pthat targets the Universal device family.

So go back and check your apps. If you consume an extension SDK, such as the Desktop extension, then you have two choices. Either mark your program as requiring a Desktop device, or add checks to your app before it tries to use Desktop-only features.

Bonus chatter: Naturally, I would prefer that you add the checks to the app so that it will also run on Windows 10X. You can test your app on the Windows 10X emulator to validate that the feature-detection works correctly.

Raymond Chen

Follow