# The activeCodePage manifest element can be used for more than just setting UTF-8 as the active code page

devblogs.microsoft.com/oldnewthing/20220531-00

May 31, 2022

Raymond Chen

Windows 10 Version 1903 introduced a new manifest property called `activeCodePage`, which you can use to set the process's code page to UTF-8.

Starting in Windows Server 2022 (and Windows 11),[1] you can also use this property to pick non-UTF-8 code pages in unpackaged app manifests.[2]

AppLocale finally grows up!

If you specify an `activeCodePage` of `Legacy`, then you're saying that your application actively hates UTF-8. It will be given the system default locale, provided a system default local exists and it isn't UTF-8. In those exceptional cases, you get code page 1252 as a fallback.

You can also specify a locale name such as `en-US` or `ja-JP`, and the application gets the code page corresponding to that locale. If the locale is not recognized by the system, then you get UTF-8 as a fallback.

This trick is most useful for application compatibility purposes: If you have some legacy application that assumes a specific locale, you can deploy a manifest next to the application that says "I don't care what locale the system is using. Give this application the locale it wants."

Here's a summary table, since I know people love summary tables.

| **activeCodePage** | `UTF-8` | specific locale | `Legacy` |
|---|---|---|---|
| **Case-sensitive?** | Yes | No | Yes |
| **Preferred result** | UTF-8 | That locale, if available | System locale, if available, and not UTF-8 |

| | | | |
|---|---|---|---|
| **Fallback result** | | UTF-8 | 1252 |
| **Packaged app** | Supported | Not supported | Not supported |
| **Unpackaged app** | Supported | Supported | Supported |

[1] Specifically, build 20348 and higher, so that also includes HoloLens 2, also known as Windows Holographic, version 21H1.

[2] The code name for unpackaged app manifests[3] was “fusion”, so if you read the documentation, you will frequently see them referred to as “fusion manifests”.

[3] They weren’t called “unpackaged app manifests” at the time, because when the fusion project was being written, we didn’t have packaged apps at all. They were just called “manifests”. Later, when manifests were introduced for packaged apps, the retronym “unpackaged app manifests” was invented to describe fusion manifests.

Raymond Chen

**Follow**