# What is this rogue version 1.0 of the HTML clipboard format?

devblogs.microsoft.com/oldnewthing/20130118-00

January 18, 2013

Raymond Chen

At least as of the time this article was originally written, the HTML clipboard format is officially at version 0.9. A customer observed that sometimes they received HTML clipboard data that marked itself as version 1.0 and wanted to know where they could find documentation on that version.

As far as I can tell, there is no official version 1.0 of the HTML clipboard format.

I hunted around, and the source of the rogue version 1.0 format appears to be the WPF Toolkit. Version 1.0 has been the version used by `ClipboardHelper.cs` since its initial commit.

If you read the code, it appears that they are not generating HTML clipboard data that uses any features beyond version 0.9, so the initial impression is that it's just somebody who jumped the gun and set their version number higher than they should have. The preliminary analysis says that you can treat version 1.0 the same as version 0.9.

But that's merely the preliminary analysis.

A closer look at the `GetClipboardContentForHtml` function shows that it generated the HTML content incorrectly. The code treats the fragment start and end offsets as character offsets, not byte offsets. But the offsets are explicitly documented as in bytes.

| | |
|---|---|
| StartFragment | Byte count from the beginning of the clipboard to the start of the fragment. |
| EndFragment | Byte count from the beginning of the clipboard to the end of the fragment. |

My guess is that the author of that helper function made two mistakes that partially offset each other.

1. The author failed to take into account that C# operates in Unicode, whereas the HTML clipboard format operates in UTF-8. The byte offset specified in the HTML format header is the byte count in the UTF-8 encoding, not the byte count in the UTF-16 encoding used by C#.
2. The author did all their testing with ASCII strings. UTF-8 has the property that ASCII encodes to itself, so one byte equals one character. If they had tested with a non-ASCII character, they would have seen the importance of the byte count. (Or maybe they simply would have gotten *more confused*.)

Now, WPF knows that the `DataFormats.HTML` clipboard format is encoded in UTF-8, so when you pass a C# string to be placed on the clipboard as HTML, it knows to convert the string to UTF-8 before putting it on the clipboard. But it doesn't know to convert the offsets you provided in the HTML fragment itself. As a result, the values encoded in the offsets end up too small if the text contains non-ASCII characters. (You can see this by copying text containing non-ASCII characters from the DataGrid control, then pasting into Word. Result: Truncated text, possibly truncated to nothing depending on the nature of the text.)

There are two other errors in the `GetClipboardContentForHtml` function: Although the code attempts to follow the recommendation of the specification by placing a `<!--EndFragment-->` marker after the fragment, they erroneously insert a `\r\n` in between. Furthermore, the EndHTML value is off by two. (It should be `DATAGRIDVIEW_htmlEndFragment.Length`, which is 38, not 36.)

Okay, now that we see the full situation, it becomes clear that at least five things need to happen.

The immediate concern is what an application should do when it sees a rogue version 1.0. One approach is to exactly undo the errors in the WPF Toolkit: Treat the offsets as character offsets (after converting from UTF-8 to UTF-16) rather than byte offsets. This would address the direct problem of the WPF Toolkit, but it is also far too aggressive, because there may be another application which accidentally marked its HTML clipboard data as version 1.0 but which does not contain the exact same bug as the WPF Toolkit. Instead, applications which see a version number of 1.0 should treat the EndHTML, EndFragment, and EndSelection offsets as untrustworthy. The application should verify that the EndFragment lines up with the `<!--EndFragment-->` marker. If it does not, then ignore the specified value for EndFragment and infer the correct offset to the fragment end by searching for the last occurrence of the `<!--EndFragment-->` marker in the clipboard data, but trim off the spurious `\r\n` that the WPF Toolkit erroneously inserted, if present. Similarly, EndHTML should line up with the end of the `</HTML>` tag; if not, the specified offset should be ignored and the correct value inferred. Fortunately, the WPF Toolkit does not use EndSelection, so there is no need to attempt to repair that value, and it does not use multiple fragments, so only one fragment repair is necessary.

Welcome to the world of application compatibility, where you have to accommodate the mistakes of others.

Some readers of this Web site would suggest that the correct course of action for your application is to detect version 1.0 and put up an error message saying, “The HTML on the clipboard was placed there by a buggy application. Contact the vendor of that application and tell them to fix their bug. Until then, I will refuse to paste the data you copied. Don’t blame me! I did nothing wrong!” Good luck with that.

Second, the authors of the WPF Toolkit should fix their bug so that they encode the offsets correctly in their HTML clipboard format.

Third, at the same time they fix their bug, they should switch their reported version number back to 0.9, so as to say, “Okay, everybody, this is the not-buggy version. No workaround needed any more.” If they leave it as 1.0, then applications which took the more aggressive workaround will end up double-correcting.

Fourth, the maintainers of the HTML clipboard format may want to document the rogue version 1.0 clipboard format and provide recommendations to applications (like I just did) as to what they should do when they encounter it.

Fifth, the maintainers of the HTML clipboard format must not use version 1.0 as the version number for any future revision of the HTML clipboard format. If they make another version, they need to call it 0.99 or 1.01 or something different from 1.0. Version 1.0 is now tainted. It’s the version number that proclaims, “I am buggy!”

At first, we thought that all we found was a typo in an open-source helper library, but digging deeper and deeper revealed that it was actually a symptom of a much deeper problem that has now turned into an industry-wide five-pronged plan for remediation.

Raymond Chen

**Follow**