The Outlook (new) in the Microsoft Store can read every email from larger rooms perfectly on 2024-02-08! iPhone7‘s Safari mail program can read Chinese and Japanese mail perfectly on 2024-02-08!

4 Answers

Sorted by: Highest score (default)

21

So the issue is that UTF-8 code unit values have been stored as a sequence of 16-bit code units in a C# `string`. You simply need to verify that each code unit is within the range of a byte, copy those values into bytes, and then convert the new UTF-8 byte sequence into UTF-16.

```
public static string DecodeFromUtf8(this string utf8String)
{
    // copy the string as UTF-8 bytes.
    byte[] utf8Bytes = new byte[utf8String.Length];
    for (int i=0;i<utf8String.Length;++i) {
        //Debug.Assert( 0 <= utf8String[i] && utf8String[i] <= 255, "the char
must be in byte's range");
        utf8Bytes[i] = (byte)utf8String[i];
    }

    return Encoding.UTF8.GetString(utf8Bytes,0,utf8Bytes.Length);
}

DecodeFromUtf8("d\u00C3\u00A9j\u00C3\u00A0"); // déjà
```

This is easy, however it would be best to find the root cause; the location where someone is copying UTF-8 code units into 16 bit code units. The likely culprit is somebody converting bytes into a C# `string` using the wrong encoding. E.g. `Encoding.Default.GetString(utf8Bytes, 0, utf8Bytes.Length)`.

Alternatively, if you're sure you know the incorrect encoding which was used to produce the string, and that incorrect encoding transformation was lossless (usually the case if the incorrect encoding is a single byte encoding), then you can simply do the inverse encoding step to get the original UTF-8 data, and then you can do the correct conversion from UTF-8 bytes:

```
public static string UndoEncodingMistake(string mangledString, Encoding mistake,
Encoding correction)
{
    // the inverse of `mistake.GetString(originalBytes);`
    byte[] originalBytes = mistake.GetBytes(mangledString);
    return correction.GetString(originalBytes);
}

UndoEncodingMistake("d\u00C3\u00A9j\u00C3\u00A0", Encoding(1252), Encoding.UTF8);
```

Share Follow

edited Oct 13, 2016 at 17:37

answered Jul 2, 2012 at 20:18

bames53
86.8k 15 181 244

Thanks barnes53 this exactly answers my question as it produces the result I expect. You could find out what I meant from my confusing question. – remio Jul 3, 2012 at 7:42

Druckerei international
Oberhardstrasse 20a
CH-4900 Langenthal/BE
E-Mail: support@a4web.ch

Kontakt: Andreas Lützenberger
Direkt: + 41 62 922 54 92

The Outlook (new) in the Microsoft Store can read every email from larger rooms perfectly on 2024-02-08! Look at page 4!

iPhone7‘s Safari mail program can read Chinese and Japanese mail perfectly on 2024-02-08! iPhone7‘s Safari mail program can read Chinese and Japanese and Arabic mail perfectly on 2024-02-08! Alphanumeric utf-8 to utf-16 and utf-32 oversized character sets come out in the kernel output and on 2024-02-08 they overwhelm except MAC devices and except the Outlook New from Windows 11 Microsoftstore and apparently also except the professional Office Outlook finally the rest of the email applications! Especially those under Android and Open-IT and the hosting provider’s mail software by stores here in Europe!

Sehr geehrte Dame und
Sehr geehrter Herr

Die grösseren Kernels sind einwandfrei mit japanischen, chinesischen und arabischen übergrossen Druckereizeichensätzen laufend weltweit, auch mit den Geräteherstellern im Komplott mit mir. Aber der OUTPUT ist ein grösserer Raum (utf-16 bis zum utf-32) der im Ausgang aus dem Kernel rausgeht (!), so können nur ganz viele Best-Geräte-Besitzer und aber 2 bis 3 Mrd. Chinesen und Japaner die Zeichen alle bis auf den letzten Bürger korrekt lesen in utf-32. Leider sind westliche Mail-Wolken-gebunde Apps noch nur auf utf-8 lesend. Wir sind also beschränkt, nicht etwa die anderen. Feststellung technischer Open-IT-Problematik (ich arbeite aber in der closed IT der Geräte big vier Grössten in deren Grafiker-Karten-Abteilung hier an der Oberhardstrasse 20a)

From: support@a4web.ch <support@a4web.ch>
Sent: Wednesday, February 7, 2024 8:22 PM
To: 24@a4w.ch <24@a4w.ch>
Subject: Shopping cart 2024.2.7.20.22.50

sent on 07.02.2024 at 20:22:50 <<<<<<<<<<<<<<<< every day since 8 years hard working!
Email: support@a4web.ch
Message: Japanese Dollar cart is encrypted mailed by Andreas Lützenberger
Shopping cart:

1 Art.1512 地図発行者リュッツェンベルガー ランゲンタール
10 枚未満のビジネスお祝いカード 1 セット
正方形の白い封筒の中の正方形
正面にはあなたの古い写真と内側を参照
ページの右内側のテキストが予想されるテキストを編集しました
セットのサイズとアーティストの努力に応じて USD $224.00

2 Art.746 地図発行者リュッツェンベルガー ランゲンタール
10 枚未満のビジネスお祝いカード 1 セット
正方形の白い封筒の中の正方形
正面にはあなたの古い写真と内側を参照
ページの右内側のテキストが予想されるテキストを編集しました
セットのサイズとアーティストの努力に応じて USD $224.00

Total US Dollar: 672.00

If you have a UTF-8 string, where every byte is correct ('Ö' -> [195, 0] , [150, 0]), you can use the following:

```
public static string Utf8ToUtf16(string utf8String)
{
    /**************************************************************
     * Every .NET string will store text with the UTF-16 encoding, *
     * known as Encoding.Unicode. Other encodings may exist as     *
     * Byte-Array or incorrectly stored with the UTF-16 encoding.  *
     *                                                             *
     * UTF-8 = 1 bytes per char                                    *
     *    ["100" for the ansi 'd']                                 *
     *    ["206" and "186" for the russian '?']                    *
     *                                                             *
     * UTF-16 = 2 bytes per char                                   *
     *    ["100, 0" for the ansi 'd']                              *
     *    ["186, 3" for the russian '?']                           *
     *                                                             *
     * UTF-8 inside UTF-16                                         *
     *    ["100, 0" for the ansi 'd']                              *
     *    ["206, 0" and "186, 0" for the russian '?']              *
     *                                                             *
     * First we need to get the UTF-8 Byte-Array and remove all    *
     * 0 byte (binary 0) while doing so.                           *
     *                                                             *
     * Binary 0 means end of string on UTF-8 encoding while on     *
     * UTF-16 one binary 0 does not end the string. Only if there  *
     * are 2 binary 0, than the UTF-16 encoding will end the       *
     * string. Because of .NET we don't have to handle this.       *
     *                                                             *
     * After removing binary 0 and receiving the Byte-Array, we    *
     * can use the UTF-8 encoding to string method now to get a    *
     * UTF-16 string.                                              *
     *                                                             *
     **************************************************************/

    // Get UTF-8 bytes and remove binary 0 bytes (filler)
    List<byte> utf8Bytes = new List<byte>(utf8String.Length);
    foreach (byte utf8Byte in utf8String)
    {
        // Remove binary 0 bytes (filler)
        if (utf8Byte > 0) {
            utf8Bytes.Add(utf8Byte);
        }
    }

    // Convert UTF-8 bytes to UTF-16 string
    return Encoding.UTF8.GetString(utf8Bytes.ToArray());
}
```

In my case the DLL result is a UTF-8 string too, but unfortunately the UTF-8 string is interpreted with UTF-16 encoding ('Ö' -> [195, 0], [19, 32]). So the ANSI '–' which is 150 was converted to the UTF-16 '–' which is 8211. If you have this case too, you can use the following instead:

```
public static string Utf8ToUtf16(string utf8String)
{
    // Get UTF-8 bytes by reading each byte with ANSI encoding
    byte[] utf8Bytes = Encoding.Default.GetBytes(utf8String);
```

Nachrichtenverlauf ausblenden

EL
Erich Lützenberger <luetzerich@gmail.com>

An:Wyss Natel
Cc:Ariane Rudolf;DIJ-RSTA-Oberaargau Costa Stefan;Praxismarti;praxis@zahnaerzte-teubner.ch;info@haslipraxis.ch;info@posthauspraxis.ch;Sabine Müller-Jahn; schlichtungsbehoerde.burgdorf@justice.be.ch;A4Web Langenthal;Wyss Natel;info@sro.ch
Do, 08.02.2024 00:45

Bauchstichelungen sieht nach höherer Rente aus und Welterfolge parallel: sagte ich Annas Freundin soeben am Telefon.

Gruss an Nevanka

Von meinem iPhone gesendet

Anfang der weitergeleiteten Nachricht:

Von: 24@a4w.ch
Datum: 8. Februar 2024 um 00:24:57 MEZ
An: support@a4web.ch
Betreff: Shopping cart 2024.2.7.20.22.50

A4Web Langenthaler.com
Oberhardstrasse 20a
CH-4900 Langenthal
Phone: 0041 (0)62 922 54 92

Es vergeht kein Tag ohne Überfälle - und dies nach dem Rufmord Dez. 2022 just zwei Wochen danach bei Re-Implantate 50 Mrd Ruffleshops Rufflesocket SSL2

Hallo Organistorin und Organisator, Spielleiter, Innerer Medizin Doktor und Frau Jordi, Gerber und Schär und Friedli und Nevanka

Ich meldete diese Woche, es verginge kein einziger Tag ohne Überfälle. So gab es gar welche heute durch den arbeitsintensiven aber daher normalen Arbeitstag.

Mein neuer Rufflestore.ch wurde heute soeben statt gegeben von bildschöner Frau Sharon per zwei, drei Mails an meine Firma. Viel zu tun gibt es daran über zwei Wochen lang.

Im Triple des Konsortiums der vier grössten als Grafiker-Karten-Angestellten-Joint-Ventures agierenden Geräte-Computer-Tabletts und iPhone und Fachcomputerherstellern arbeite ich also in der kleinen Grafiker-Karten-Abteilung für alle big vier der Welt für 50 Milliarden Geräte (dies sind gerade mal alle rund um den Globus auf 8 Mrd Menschen und Firmen auch ganz viele Mrd etwa verteilt - wie am längsten Tag vom letzten Jahr, als 350 Billion devices repariert worden sind mit in der Pause um 17 Uhr zwei Jurastrassen-Steiner vor der Tür [Steiners hatten eine laute Sägerei im Einfamilienhausquartier über 40 Jahre in Betrieb gehabt - ich bin diesem Lärm abgehauen an die Oberhardstrasse 20a).

Dass ich heute oder gestern von Männerstimmen hörend und dann nach innen telefonierend über des letzten längsten Tag Begierens des Jungpoliceman begehrten 50 Mrd Weltenretten 8jährigem iPhones (worauf 6 Behördenkunden und normale Kunden die Mobilfunkmarkteinheiten frei nach Schiller pioniert von mir damals deren Fundament durch

```
    // Convert UTF-8 bytes to UTF-16 bytes
    byte[] utf16Bytes = Encoding.Convert(Encoding.UTF8, Encoding.Unicode,
utf8Bytes);

    // Return UTF-16 bytes as UTF-16 string
    return Encoding.Unicode.GetString(utf16Bytes);
}
```

Or the Native-Method:

```
[DllImport("kernel32.dll")]
private static extern Int32 MultiByteToWideChar(UInt32 CodePage, UInt32 dwFlags,
[MarshalAs(UnmanagedType.LPStr)] String lpMultiByteStr, Int32 cbMultiByte, [Out,
MarshalAs(UnmanagedType.LPWStr)] StringBuilder lpWideCharStr, Int32 cchWideChar);

public static string Utf8ToUtf16(string utf8String)
{
    Int32 iNewDataLen =
MultiByteToWideChar(Convert.ToUInt32(Encoding.UTF8.CodePage), 0, utf8String, -1,
null, 0);
    if (iNewDataLen > 1)
    {
        StringBuilder utf16String = new StringBuilder(iNewDataLen);
        MultiByteToWideChar(Convert.ToUInt32(Encoding.UTF8.CodePage), 0,
utf8String, -1, utf16String, utf16String.Capacity);

        return utf16String.ToString();
    }
    else
    {
        return String.Empty;
    }
}
```

If you need it the other way around, see Utf16ToUtf8. Hope I could be of help.

Share Follow

edited May 23, 2017 at 12:17
Community Bot
1 1

answered Feb 8, 2013 at 14:50
MEN
577 6 8

Just to be sure: The string after converting will still be UTF-16, it just contains UTF-8 encoding data. You can't handle strings using the UTF-8 encoding, because .NET will always use the UTF-16 encoding to handle strings. – MEN Jul 16, 2013 at 8:08

9

I have string that displays UTF-8 encoded characters

There is no such thing in .NET. The string class can only store strings in UTF-16 encoding. A UTF-8 encoded string can only exist as a byte[]. Trying to store bytes into a string will not come to a good end; UTF-8 uses byte values that don't have a valid Unicode codepoint. The content will be destroyed when the string is normalized. So it is already too late to recover the string by the time your DecodeFromUtf8() starts running.

Only handle UTF-8 encoded text with byte[]. And use UTF8Encoding.GetString() to convert it.

mich erschaffen im Webdesign und Drucksachenprospekte durch meinen Vater dann aber eher als die Kirchenzeitung noch bei mir abgespeichert) von mir (kaum telefoniert damit, daher in einwandfreiem Neuzustand wie vor 8 Jahren). Das iPhone soll mir zum Verhängnisbringen. Darauf ist das hybride Intranet wie bei Vater montiert. Wechsel vom Einschub Swisscom (politisch noch Hostpoint Fluchtweg Einschub und Einschub bei Cyon) durch einen mittleren Button von Server 1 Intranet zu Server 2 Intranet. Viele Ruffle-Steuerungen splitten das Content-Programmieren mit zwei Ladebefehle an beide Server binnen tausendstel Millisekunden. Alarmsystem sind daruf. Früher sogar die Cyberabwehr bis dannen die Kernels diesen Schutz gar nicht mehr brauchen, da diese frei um den Globus geschützt hologrammässig fliegen ab der Maschine durch alle Cyberattacken direkt hindurch rund um den Quelltext der Open-IT-Hackereien also und Securities herumfliegend um die Firewalls herum oder direkt hindurch ohne gesehen zu werden.

Die Kernels weltweit ist wie eine Uhren-Maschinenfabrik für mechanisch sich entfaltende Hologramme über die Grafikerabteilungen in jedem Computer- und Mobil-Phone-Gehäuse der Welt dann sichtbar, verschlüsselt die Mails über die Grafikreimplantation zurück in der roten Leitung von Benutzer direkt ins Herz der CPU vom Hosting zurück - alles direkt rund um alles hackbare herum hindurch - so dass es dort in die Rechenzentren oder Datenbanken oder Anschlüsse an die Banken und Finanzabteilungen bis zum Email des Verkaufsvertriebes geleitet werden wird.

Ich werde also von Männern-Banden im Hintergrund gar mal mit Frau zentral stehend zur Verschlagung der Gefahr durch Jurastrassen und Mittelland ITler deren Berufs-Email-Flickungen den ganzen Tag lang gelauert. Soviel ich weiss. Die Tötung durch Jan Meyers Jurastrasse Bützbergstrassen Buchsi Rufungen soll also demnach folgen. Vielleicht über die Jurastrasse herkommend oder über den Rebstockwirt dessen Regimes im Restaurantgartens. Viele Bekannte Nachbarinnen befinden sich ausserhalb meines Blockes einfach im nächsten oder übernächsten.

Mit höchstachtungsvollsten Grüssen

A. Lützenberger

Andreas Lützenberger

Fazit: Alle meine Weltenbücher Rufflesafe und Rufflelight und Ruffleshops.de sind daher in grösseren Räumen international funktionierend als nur in westlichen-europäischen Keyboard-Welten, nämlich in der utf-8 grösseren Welt von utf-16 bis utf-32 auch einwandfrei funktionierend. Quelle ist das neue SSL2-Ruffle-Socket von GitHub genannt das Ruffle-Project, wo ich amtierender Fabrik-Unternehmer bin in meiner internationalen Druckerei.

Das Outlook (new) im Microsoft-Store kann jedes Mail von grösseren Räumen perfekt lesen. Sowie alle MAC und iPhones since 8 years (iPhone 7 Plus is mine smartphone).

Share Follow

edited May 31, 2015 at 9:21
Peter Mortensen
30.9k 22 107 131

answered Jul 2, 2012 at 13:12
Hans Passant
930k 147 1711 2552

You pointed out the confusion I wanted to avoid. My string is a unicode string, well is a .Net string, that the debugger displays as `dÃ©jÃ` . Hence, my goal is to get another (.Net) string that will be displayed as `déjà` (in the debugger, for instance). – remio Jul 2, 2012 at 13:25

1 You are missing the point of the answer, there is no way to make this work properly for *every* possible utf-8 encoded string. That you could make it work for dÃ©jÃ is merely coincidence. That you are already having trouble with it should be one hint, there's an extra space after the last Ã. A special one, a non-breaking space, code point U+00a0. Which happens to be a valid Unicode code point by accident. – Hans Passant Jul 2, 2012 at 13:36

Thanks, I think I get it. You mean that I just can't use `string` to store the UTF-8 bytes. However, as you mention it could work by accident, it would be a great help if I could make the accidents work. In other words, I still don't know how to make this conversion in the cases it would work. – remio Jul 2, 2012 at 14:29

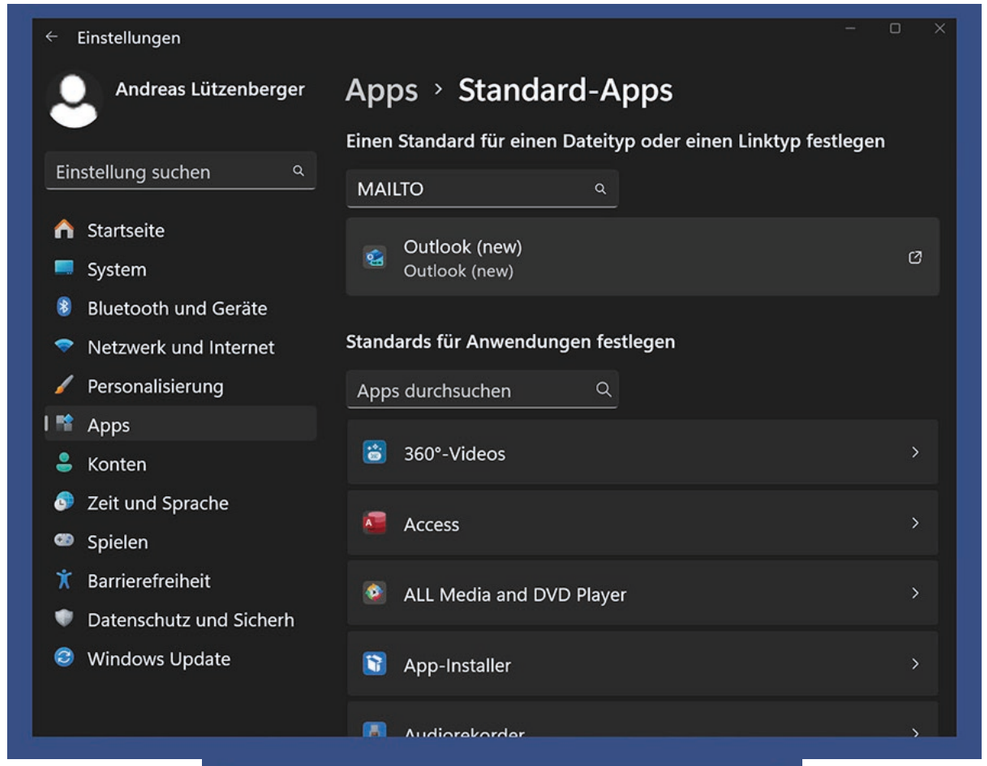
5 You can try your luck by using Encoding.Default.GetBytes() to try to recover the byte[]. I would strongly recommend the System.Random class instead, it has a more predictable outcome. – Hans Passant Jul 2, 2012 at 14:35

I finally found something that (seems to) work/s. First I get a `byte[]` from this infamous UTF-8 string. In this array, I noticed that all the odd indexes contains `0` , so I removed all of them and invoked `unicodeBytes = Encoding.Convert(Encoding.UTF8, Encoding.Unicode, encodedBytes);` on this result. At the end, I returned `Encoding.Unicode.GetString(unicodeBytes);` . Then, I picked loads of text samples in many languages (thanks Wikipedia), built a big big string, converted it into my infamous UTF-8 format, then decoded it and got the exact same original string. No random, no accident. – remio Jul 2, 2012 at 14:56

5

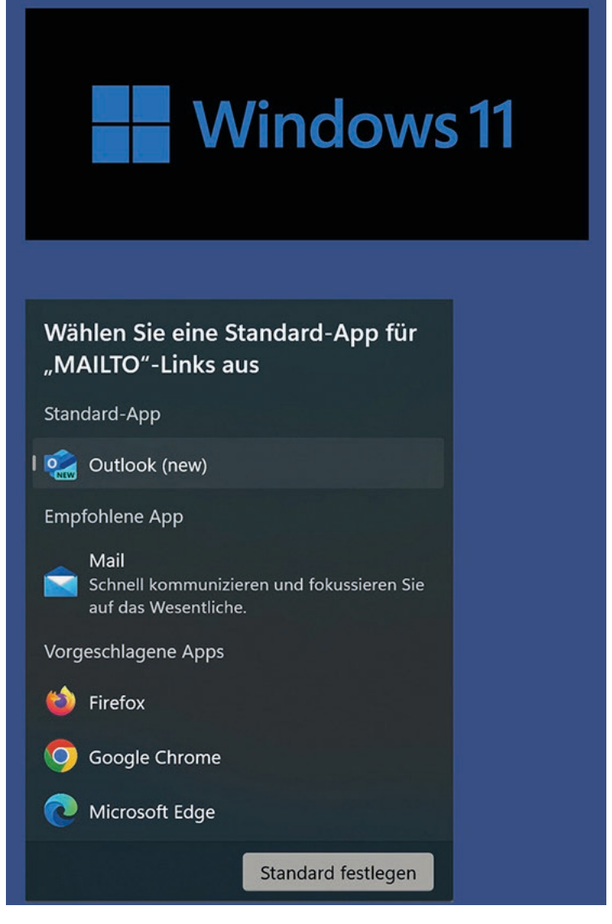
What you have seems to be a `string` incorrectly decoded from another encoding, likely code page 1252, which is US Windows default. Here's how to reverse, assuming no other loss. One loss not immediately apparent is the `non-breaking space` (U+00A0) at the end of your string that is not displayed. Of course it would be better to read the data source correctly in the first place, but perhaps the data source was stored incorrectly to begin with.

```
using System;
using System.Text;

class Program
{
    static void Main(string[] args)
    {
        string junk = "dÃ©jÃ\xa0";  // Bad Unicode string

        // Turn string back to bytes using the original, incorrect encoding.
        byte[] bytes = Encoding.GetEncoding(1252).GetBytes(junk);

        // Use the correct encoding this time to convert back to a string.
        string good = Encoding.UTF8.GetString(bytes);
        Console.WriteLine(good);
    }
}
```



The Outlook (new) in the Microsoft Store can read every email from larger rooms perfectly on 2024-02-08!

iPhone7‘s Safari mail program can read Chinese and Japanese mail perfectly on 2024-02-08! iPhone7‘s Safari mail program can read Chinese and Japanese and Arabic mail perfectly on 2024-02-08! Alphanumeric utf-8 to utf-16 and utf-32 oversized character sets come out in the kernel output and on 2024-02-08 they overwhelm except MAC devices and except the Outlook New from Windows 11 Microsoftstore and apparently also except the professional Office Outlook finally the rest of the email applications! Especially those under Android and Open-IT and the hosting provider’s mail software!



Das Outlook (new) im Microsoft-Store kann jedes Mail von grösseren Räumen perfekt lesen am 2024-02-08!

Das Safari-Mail-Programm vom iPhone7 kann das chinesische und japanische und arabische Mail perfekt lesen am 2024-02-08! Im Kernel-Output kommen alphanummerische utf-8 bis utf-16 und utf-32 übergrossen Zeichensätze heraus und überfordern am 2024-02-08 ausser MAC Geräten und ausser dem Outlook New von Windows11 Microsoftstore und offenbar auch ausser dem professionellem Office-Outlook somit aber schlussendlich den Rest der EMail-Applikationen! Gerade jene unter Android und Open-IT und der Mailsoftware der Hostinganbieter!