The text/nfo Media Type
draft-seantek-text-nfo-03

Abstract

This document registers the text/nfo media type for use with release iNFOrmation. While compatible with text/plain, ".NFO" files and content have distinguishing characteristics from typical plain text because they are meant to be output to IBM PC-compatible system consoles that support certain "ANSI" escape sequences.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. iNFOrmation

Packagers of files or other bundled content commonly include a common human-readable manifest that describes their packages. While an obvious solution is to include a README, in an archive such as a ZIP file, READMEs are generally written for software applications and provide late-breaking instructions on how to configure and install the software, along with known bugs and changelogs. (Plain) text READMEs are also generally limited to printable US-ASCII characters.

Starting from circa 1990, packagers of various types of content settled upon the Release iNFOrmation format (NFO, commonly pronounced "EN-foe" or "info") to describe their releases. An NFO file serves similar purposes to a README, but with several nuanced differences. NFOs usually contain release information about the media, rather than about software per-se. NFOs credit the releasers or packagers. Much like the Received: Internet Message header [RFC5322], intermediates ("couriers") can also insert NFOs.

Most distinctively, NFOs have come to contain elaborate ASCII or ANSI artwork that is remarkable in its own right in the pantheon of the postmodern computing culture. Many NFOs have been authored with the intent of displaying them on a terminal display with monospaced, inverted text (black background, gray or off-white foreground); some NFOs even include escape sequences to generate animations or color. The widely accepted encoding for NFOs is "OEM Code Page 437", the character set of the original IBM PC and MS-DOS.

When served in the same manner as plain text (text/plain), a lot of the elaborate artwork in NFOs is lost, garbled, or misaligned on display. As NFOs are still in considerable use, the goal of this registration is to rectify these interchange problems and reclaim this piece of living computer history.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Release iNFOrmation Media Type Registration Application

Type name: text

Subtype name: nfo

Required parameters:

charset: Per Section 4.2.1 of [RFC6838], charset is REQUIRED. Unlike most other text types, the default value is the character set of
the original IBM PC and MS-DOS, called OEM Code Page 437, and named "oem437". Implementations MUST support OEM Code Page 437. Unfortunately, the simple application of the IANA registered character set "IBM437" (aka "cp437") [RFC1345] will miss some important characters, so conformant implementations MUST support OEM Code Page 437 as specified in Section 3. NFOs authored for more modern computing environments are known to use ISO-8859-1, ISO-8859-15 (including support for the Euro sign), or UTF-8; however, for maximum interoperability, these or any other character sets MUST be declared by the sender. When absent, a receiver MAY guess, but SHOULD heavily bias the outcome towards OEM Code Page 437 unless UTF-8 encoding is patently obvious. A RECOMMENDED detection algorithm is provided in Appendix A.

Optional parameters:

baud: A natural number (integer greater than 0) indicating the gross bit rate ("symbol rate") at which the NFO is supposed to be rendered to screen. This optional parameter provides a nostalgic effect from the days of dialup modems and fixed-speed serial lines. It also controls the animation rate, to the extent that the NFO employs optional escape sequences. While the term "bps" might be more accurate, this parameter is meant to be interpreted the way an end user would experience the real-world conditions that a dialup modem would provide on the eve of Y2K. (The term "baud" is also used by a couple of popular modern viewers of this format.) For example, a conforming implementation could implement "57600" as if the data were being downloaded using a V.92 modem, replete with random stalls due to retransmission attempts on account of noise on the line.

Encoding considerations:

Text with 8-bit code points; all 8-bit combinations (including NUL) are possible.

Security considerations:

It’s just text; this format provides no facilities for confidentiality or integrity. The ANSI escape sequence "CSI 5 m" could, however, blink you to death. As only a subset of ANSI escape sequences MUST be interpreted; interpreting a greater range than the subset prescribed in this registration may introduce other security issues, such as transmitting operating system commands.

Some code points in oem437 have been used ambiguously in practice, so implementations SHOULD NOT assume that the mapping between this charset and Unicode is bijective. When displayed, codes 00, 20, and
FF MAY appear to be similar, i.e., as a blank space.

Interoperability considerations:

NFOs are plain text but look best when read in a terminal view or with a dedicated NFO viewer that can emulate terminal features. As a result, they SHOULD be treated differently than text/plain files. The reference environment for NFO viewers to emulate is an IBM PC-compatible machine running MS-DOS 6.22 with the ANSI.SYS MS-DOS device driver loaded, where the NFO is displayed as if it were output to the terminal using the "TYPE" command.

Published specification: [[Note to RFC Editor: Insert number here.]]

Applications that use this media type:

NFO viewers; text editors; terminals.

Fragment identifier considerations:

Same as text/plain [RFC5147].

Additional information:

Deprecated alias names for this type: text/x-nfo

File extension(s): .nfo

Macintosh file type code(s):

TEXT. A uniform type identifier (UTI) of "public.nfo", which conforms to "public.plain-text", is RECOMMENDED.

Person & email address to contact for further information:

Sean Leonard <dev+ietf@seantek.com>

Restrictions on usage: None.

Author/Change controller: Sean Leonard <dev+ietf@seantek.com>

Intended usage: COMMON

Provisional registration? No

3. OEM Code Page 437

"OEM Code Page 437" refers to the character set of the original IBM PC and MS-DOS. The code page actually represents two related things: the set of 256 graphemes stored in video read-only memory (ROM) that are accessed with a single 8-bit code, and an 8-bit encoding for text...
content that displays the graphemes or causes other behavior as defined by the code, the operating system, and the loaded device drivers. NFO is encoded with the aforementioned 8-bit encoding, which means that not all 256 graphemes are directly available for use.

For example: the sequence 0D 0A (CR LF) identifies a new line; the code 1A (SUB) is the MS-DOS end-of-file marker. The code 0D cannot be used directly to express the grapheme U+266A EIGHTH NOTE; the code 0A cannot be used directly to express the grapheme U+25D9 INVERSE WHITE CIRCLE; the code 1A cannot be used to express U+2191 RIGHTWARDS ARROW.

The registration for IBM437 [RFC1345] is used as a basis for this specification, which only elaborates upon the differences. Suggested mappings to Unicode characters are included; however, the mapping is not bijective. Octets are in hexadecimal. The symbols below next to the octets match [RFC1345], although the actual character has the meaning described here rather than the [RFC1345] meaning.

3.1. Low-Order Codes (00-7F)

The codes in the 20-7E range are the same as in US-ASCII and IBM437.

01-06, 0B, 0C, 0E-19, and 1C-1F are displayed as their corresponding ROM graphemes.

00 NUL is displayed (and treated) as a space. Depending on the output environment, an implementation MAY map this code to U+0000 NULL, or U+0020 SPACE.

07 BEL MAY cause an audible bell sound (beep) to be emitted. Actually emitting a sound is not required for conformance. However, implementations that progressively render the output MUST pause for this code as if a sound were emitted.

08 BS causes the prior character to be erased: the prior grapheme is displayed and treated as a regular or non-breaking space (SP or NBSP), depending on whether the prior character would have been breaking or non-breaking.

09 HT causes horizontal tabbing, which for purposes of conformance, SHOULD produce the equivalent spaces so that the subsequent text is aligned on the next 8-character boundary.

0A LF causes a new line to be created and the text insertion point ("cursor") to be moved to the beginning of that line.

0D CR causes the text insertion point ("cursor") to be moved to the
beginning of the current line. Subsequent text will overwrite
the characters on the current line, until the cursor moves
somewhere else. (0A creates and moves the cursor to a new
line; therefore, 0A in the middle of overwriting the current
line will not insert or erase any characters that might
otherwise be on that line.)

1A SUB is the MS-DOS end-of-file (EOF) marker; it ends the display.
Codes after 1A MUST NOT be displayed. 1A can be used to
delimit metadata from the main NFO content, although this
practice is rarely used for NFOs. A well-known metadata format
in this technology area is SAUCE (Standard Architecture for
Universal Comment Extensions) [SAUCE], which implementations
MAY support. A SAUCE record can specify a different code page.
An implementation that supports SAUCE SHOULD support following
the code page directive in the SAUCE record when the MIME
entity’s charset is oem437.

1B ESC may be the start of an ANSI ESC sequence. If no valid ESC
sequence is recognized, output the corresponding ROM grapheme
(U+2190 LEFTWARDS ARROW) and continue normal processing with
the next code.

7F DEL is displayed as the corresponding ROM grapheme (U+2302 HOUSE).

3.2. High-Order Codes (80-FF)

The codes in the 80-AF range are a selection of Latin characters;
they are the same as in IBM437. A conformant implementation MUST NOT
treat these codes as C1 control characters.

The codes in the B0-DF range are box drawing and block characters;
they are the same as in IBM437.

The codes in the E0-FF range are for mathematical symbols, which are
the same as in IBM437, with the following exceptions. The preferred
Unicode mapping in Microsoft’s OEM Code Page 437 documentation is
designated with [OEMCP437]:

E1 b* can be either U+03B2 GREEK SMALL LETTER BETA, or U+00DF LATIN
SMALL LETTER SHARP S (German Eszett) [OEMCP437]. The two were
undistinguishable at low resolution on the original IBM
hardware. Newer grapheme sets, including those of the IBM EGA
and VGA graphics cards, display this code as the Eszett.
Unfortunately only context can determine the proper character
to use.

E3 p* can be U+03C0 GREEK SMALL LETTER PI [OEMCP437], U+03A0 GREEK
CAPITAL LETTER PI, or \( \U+220F \) N-ARY PRODUCT, depending on the particular grapheme used.

E4 \( \Sigma \) can be either \( \U+03A3 \) GREEK CAPITAL LETTER SIGMA [OEMCP437] or \( \U+2211 \) N-ARY SUMMATION.

E6 \( \mu \) can be either \( \U+00B5 \) MICRO SIGN [OEMCP437] or \( \U+03BC \) GREEK SMALL LETTER MU.

EA \( \Omega \) can be either \( \U+2126 \) OHM SIGN or \( \U+03A9 \) GREEK CAPITAL LETTER OMEGA [OEMCP437].

EB \( \delta \) is \( \U+03B4 \) GREEK SMALL LETTER DELTA [OEMCP437]. However, it can be used as a surrogate for \( \U+00F0 \) LATIN SMALL LETTER ETH (Icelandic, Faroese, Old English, IPA) or \( \U+2202 \) PARTIAL DIFFERENTIAL.

ED \( \emptyset \) is \( \U+03C6 \) GREEK SMALL LETTER PHI [OEMCP437], but in MS-DOS was mainly used as \( \U+2205 \) EMPTY SET. Other possible meanings include \( \U+03D5 \) GREEK PHI SYMBOL (used as a technical symbol, with a stroked glyph) (to name angles), \( \U+2300 \) DIAMETER SIGN, or \( \U+00F8 \) SMALL LETTER O WITH STROKE (as a surrogate).

EE \( \varepsilon \) is \( \U+03B5 \) GREEK SMALL LETTER EPSILON [OEMCP437] or \( \U+2208 \) ELEMENT OF.

FF \( \text{NS} \) is NBSP, also known as \( \U+00A0 \) NO-BREAK SPACE. The ROM grapheme is the same as SP (SPACE), i.e., it is blank.

### 3.3. ANSI Escape Sequences

To support NFO content containing colors and other goodies, an NFO viewer MUST support a subset of "ANSI" escape sequences. (The required sequences are not directly related to ANSI, but rather to [ANSI.SYS].)

[ANSI.SYS] supports cursor positioning, erasing, Set Graphics Mode (SGR), mode switching, and keyboard remapping. Of these functions, a conforming implementation MUST support the Set Graphics Mode (SGR) escape sequence. An implementation MUST support setting foreground colors (30-37) and background colors (40-47), which are also in [ISO6429]. An implementation MUST support all of the [ANSI.SYS] text attributes (0, 1, 4, (5 and/or 6), 7, and 8). Text attribute 5 is "Blink: Slow" (less than 150 per minute); text attribute 6 is "Blink: Fast" (more than 150 per minute). While [ANSI.SYS] does not document attribute 6, that was the behavior of the actual ANSI.SYS. An implementation SHOULD reproduce similar functionality.
The other [ANSI.SYS] escape sequences are OPTIONAL. An implementation MAY support standard or vendor-specific escape sequences. For a list of standard sequences, see, e.g., [ISO6429] and [ISO8613].

3.4. Accessing Hidden Grapheme Codes

There is no obvious way to encode the graphemes that are inaccessible at the values 07, 08, 09, 0A, 0D, 1A, and 1B. This specification provides a technique to access these graphemes in the context of OEM Code Page 437. This technique is RECOMMENDED, but not required.

Although MS-DOS and ANSI.SYS did not conform to [ISO2022], that standard defines escape sequences to switch to other character sets. Unicode contains appropriate code points for all of the inaccessible graphemes (characters). Accordingly, the escape sequence:

\texttt{ESC \% G}

switches the code to UTF-8 (with unspecified implementation level) [REG196]. While in UTF-8, the escape sequence:

\texttt{ESC \% 0}

reverts the code back to the original [ISO2022]. Normally the code would be [ISO2022], but given the starting context of OEM Code Page 437, the code returns to OEM Code Page 437. The codes are as follows:

<table>
<thead>
<tr>
<th>ROM grapheme number</th>
<th>IBM437 symbol</th>
<th>Unicode code point</th>
<th>Unicode name: UTF-8 encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>07 BEL U+2022 BULLET: E2 80 A2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08 BS U+25D8 INVERSE BULLET: E2 97 98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09 HT U+25CB WHITE CIRCLE: E2 97 8B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0A LF U+25D9 INVERSE WHITE CIRCLE: E2 97 99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0D CR U+266A EIGHTH NOTE: E2 99 AA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1A SUB U+2192 RIGHTWARDS ARROW: E2 86 92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1B ESC U+2190 LEFTWARDS ARROW: E2 86 90</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.5. UTF-8/Unicode Processing

When NFO content is encoded in UTF-8 or another Unicode encoding [UTF], the C0 and C1 code points may be present. These codes MUST be treated as control codes, not graphemes. They have the same behavior as specified for the special low-order codes described in Section...
3.1. For example, 1A ends the display, and 09 emits spaces sufficient for 8-column tabbing. 1B is ALWAYS treated as the start of an ESC sequence; if the sequence is not recognized, 1B does NOT revert to outputting a LEFTWARDS ARROW grapheme. Instead, nothing is displayed. For LEFTWARDS ARROW, encode U+2190 instead.

The C1 control code 9B (CSI: Control Sequence Introducer) (Unicode code point U+009B) MUST be recognized as such; it is equivalent to 1B 5B (ESC [).

3.6. Grapheme Reference

The following figure is a reference of all 256 graphemes in the IBM PC ROM. The figure is a MIME (base64)-encoded PNG image.

MIME-Version: 1.0
Content-Type: image/png
Content-Disposition: attachment; filename="Codepage-437.png"
Content-Transfer-Encoding: base64

iVBORw0KGgoAAAANSUhEUgAAASwAAACMCAMAAADxyGQdAAAAAGXFHWRThZ2Od2FyQ2BBZG9iZSBjWVFlJWY5c1nPAAAAAQTFREFKhioAAAAMkDP8QAACACYY24zIHzH0H/P
6XvO7daVmb1afrRjg+xuk20SWxojeBMPD5v5wRftiGVI1wdpgbbA2WJ3D/Xtkv9lq3gyWLHY3yuEHG8emOL+4Dk4b5fDm8an4YR2ud/4kJv919Ass+nXLCUARjSmN6nNc/h53yIgerj3+z+kyC1n/ECWPbYxXrKtC419j5keQQQJCVs13tfreQijW4oV7+IisAa/6d/kjUvoksksEL4MBmmfqq+igZl1vD3+nc91SC9nZ1x0UbMBpyTgdc8fp4xfXJHP4jOd4ikoHwpIFQZ011+rMeRRk8PAjGwdZBN+xP4SVa+k4GVgP32ceGAcfGZI1HlLiTLw0OscwmMaA8t2lpysP5NNSLCJmth
+uBz+j2FU66wBrVF2JqlQrl9XhD9s3kuqL1Lgrdp43KgsL6u+EsPUbJOuJYST5/1b
n00VIXTDbgwJAdoyqyhZMOXEi+hp01avYgVW5dt7E6TirRqklJrjadbDJyjIAANb6VGDhaZRXWJvU70lenWpuznakN1gbrcNcGy/42V2gPnMIS/pSwmpw3gwKEWE+M/zzijrjkwj26aRRSebuUjok7umLm7g7/7R7h2BtQj3QGPalg887ezrrJwnj3iXrXhnhoVhGlxzF7KraaDeagcwWNInZ2sfrXy5HIF6yWgX8BsaX59AcK9r+UfadsW4IFJjnK0K1pQ1ifks77/OE6p19SA+zGEB7xAvzKXKOkBoBzXupjPnFIfaXa901DDKxAzp+jxvWiEcLaaWY0DADUsN7k7CZXVDAc4suwFelliyc130K3RCrlU141WlY41glJxsvFgfpqวกxBx5WRClyOAvpcw6stPsmjBBS+vh+31gUadAf6Xko+Oy0bcYwIBwvRtewlaHH00KvrW3M3LbuxchF40e55QkVbUtbdSc5W4XgfFprz7R
dqR/jqLZYo3Hbmsldq1Wmibedyh616SDhuXSOPEDX41vaLP5CEubGANq/MobBg6tAXMBjL65xkdRanGntj1qXMVFG0405aq8mgf42zdzjbfhFGisGnPjxedIX+QxJNHM0ksXMW91ANXZ
os4jsMbaEFwNIV50+mFMcxGNCfUMTBEregVhPhjv1Agle5KLcr+HdMBbBkscH4jbebefPudmuc2wNNSGY/6y16T25eZLQzwSK4DttsESeg2ykey4mx7YpM2JMYx9wmMjCFbHgAHL
b5ciKwOrchAqkhkawachK1rsKPUKQHzxRde27TYrsFKFpCWPPXmZB2teCyGMyy9akBCDKDrSP
FhV8QAhjopN70RNhchHTBsjz2YP1bU41masXFAQoxanUqgHS2N13FNSISIsLpFEa02z0GGMkBToa
nab2lGqgSK9Z3cdehhh7D5s848RZC2O7wpmpm borderedBuZbU2zYg27Q18a83w3kItKoBPmKxeET
LrVxsQSmU0mqnFcmA34i1Le1scIzOUgW7d88BPWQAQRRFeYHPBvyMsCyI27awDkYKaph
N/IzmnHuE6Siij8Yd6qINNVDyCy9ApckjkH JVmhNkflNnfWigjKpmnsayxgJYWnVum91tITF
47ItLqp+F5Fv5EvIO1YPbaWbbbEGi0RSok4XueYzSkaLOawfddYEn+eYDG+V9kzvi1IPYw9Vx
1PPacimNrm4fa4AznBokWM6xasYCZtkUme1s9qKgkxJx8O9q/zsU6jczW8hPWyk4IzccF
+Czab8OSOH+j8nSWuLeULXDU99q1901ticH6/JxbhtVgN1Buovr+Bxtt81h85/m05vRbcEK/bRwCvVveepJ10ahvAs3YrQoiyMwkzMDqBkAEUs+HzaUghShfuxzAMqMCsYJug6Vow
3.7. Charset Registration Template

To: ietf-charsets@iana.org
Subject: Registration of new charset oem437

Charset name: oem437

Charset aliases: None.

Suitability for use in MIME text: Suitable.

Published specification(s): This specification; [OEMCP437].

ISO 10646 equivalency table:

This table is taken from the IBM437 registration in [RFC1345], with modifications based on actual implementations of [OEMCP437], as discussed in this document. Character mnemonic symbols generally map to the Unicode code points listed in Section 3 of [RFC1345], with the following exceptions. The symbol suffix $ (for example, HT$) means that the Unicode code point mapping is essentially correct, but an implementation might need to perform additional or special processing as discussed in this document, depending on the output environment.

Leonard                  Exp. December 24, 2016                [Page 10]
The symbol $$ means that this code point has special
considerations as discussed in this document, so no
single, definitive Unicode code point mapping can be given.
Finally, three characters have no corresponding mnemonic
symbols in Section 3 of [RFC1345], so symbols are defined here:

$> 25ba BLACK RIGHT-POINTING POINTER
$< 25c4 BLACK LEFT-POINTING POINTER
$B 21a8 UP DOWN ARROW WITH BASE

Additional information:

See this document for details on how to handle particular codes
that correspond both to graphemes in the IBM PC ROM, and
to control characters.

Person & email address to contact for further information:

Sean Leonard <dev+ietf@seantek.com>

Intended usage: COMMON

4. Example

The following example is a RELEASE.NFO file as an e-mail attachment,
with base64 encoding. Note that the character set is (correctly)
assumed to be OEM Code Page 437.
5. IANA Considerations

IANA is asked to register the media type text/nfo in the Standards tree using the application provided in Section 2 of this document.

IANA is asked to register the charset oem437 in the Character Sets registry using the application provided in Section 3 of this document.

6. Security Considerations

It's just text; this format provides no facilities for confidentiality or integrity. The ANSI escape sequence "CSI 5 m" could, however, blink you to death. As only a subset of ANSI escape sequences MUST be interpreted; interpreting a greater range than the subset prescribed in this registration may introduce other security issues, such as transmitting operating system commands.

Some code points in oem437 have been used ambiguously in practice, so implementations SHOULD NOT assume that the mapping between this charset and Unicode is bijective. When displayed, codes 00, 20, and FF MAY appear to be similar, i.e., as a blank space.

7. References

7.1. Normative References


[RFC5147] Wilde, E. and M. Duerst, "URI Fragment Identifiers for the


7.2. Informative References


Appendix A. IBM Code Page 437 vs. UTF-8 Detection Algorithm

In cases of ambiguity, the following algorithm SHOULD be used to detect UTF-8 encoded data in text/nfo content:

If the octets EF BB BF are present at the beginning => UTF-8.

Considering all octets in the content:
If no octets are greater than 7F => oem437.
If any octets are F5 - FF, C0, or C1 => oem437.
If any UTF-8 encodings are "ill-formed" => oem437.
If any UTF-8 encodings represent illegal code points
(e.g., surrogate code points) => oem437.

Ragged line tests:

If display characters decoded with oem437
result in identical line widths => oem437.
If display characters decoded with UTF-8
result in identical line widths => UTF-8.

Finally:
=> UTF-8 or oem437; prefer oem437.

Author’s Address

Sean Leonard
Penango, Inc.
5900 Wilshire Boulevard
21st Floor
Los Angeles, CA 90036
USA

EMail: dev+ietf@seantek.com
URI:  http://www.penango.com/