```
Network Working Group                                         S. Leonard
Internet-Draft                                             Penango, Inc.
Intended Status: Informational                         November 15, 2017
Expires: May 19, 2018
```

The text/nfo Media Type
draft-seantek-text-nfo-05

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

Copyright Notice

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 distinctly, 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 that 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 S*	can be either U+03A3 GREEK CAPITAL LETTER SIGMA [OEMCP437] or U+2211 N-ARY SUMMATION.
E6 m*	can be either U+00B5 MICRO SIGN [OEMCP437] or U+03BC GREEK SMALL LETTER MU.
EA W*	can be either U+2126 OHM SIGN or U+03A9 GREEK CAPITAL LETTER OMEGA [OEMCP437].
EB d*	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 /0	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 e*	is U+03B5 GREEK SMALL LETTER EPSILON [OEMCP437] or U+2208 ELEMENT OF.
FF 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:

```
   ESC % G
```

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

```
   ESC % @
```

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:

```
ROM grapheme number
|  IBM437 symbol
|  |   Unicode code point
|  |   |      Unicode name: UTF-8 encoding
|  |   |      |
07 BEL U+2022 BULLET: E2 80 A2

08 BS  U+25D8 INVERSE BULLET: E2 97 98

09 HT  U+25CB WHITE CIRCLE: E2 97 8B

0A LF  U+25D9 INVERSE WHITE CIRCLE: E2 97 99

0D CR  U+266A EIGHTH NOTE: E2 99 AA

1A SUB U+2192 RIGHTWARDS ARROW: E2 86 92

1B ESC U+2190 LEFTWARDS ARROW: E2 86 90
```

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
```

```
iVBORw0KGgoAAAANSUhEUgAAASwAAACMCAMAAADxyGQdAAAAGXRFWHRTb2Z0d2FyZQBBZG9i
ZSBJbWFnZVJlYWR5ccllPAAAAAZQTFRFqKioAAAAmKDP8QAACYZJREFUeNrsXYt24zoIhP//
6XvO7daVmBlAfrRJq+xuk20SWxojBMPD5vvRftiGYIO1wdpgbbA2WJ3D/Xtkv9lg3QyWLY3y
u66G8emOL+4DKz4f5yDfmo8an4YR2ud/4kjki9vAss+nXCLUCArJsmNen6hNc/fh53yIGerj
3z+kyCin6/EcWPbxYxrKcT49lJ5keQQLJCvI3tfrQeQjWG4oV7g+lsAaJd/6kjUvoxksEL4B
mmFqg+igZI1vD3+nc9FlSC9nZxlqUbMBpYtgDc8fP4xJxPHjQOd4ikoHwApiFLQZ01l+RmeR
k8PAjGwdZBn+P4SvA+K4GVggR3ZcqeGAcfGiZI1HLiTLw0rsgcWmA8ti2pIysP5NNSLCJMth
o+uBxZ+jZFU66wBrVF2JqlqQrLgl9XdD9s3kuqlliGrdp43Kgs6Su+EsPuOBiJUySTOT5/lb
n0oV1XTXdBgwWjAdoqyhZM0XIei+hp01avYgWVSdt7fD6TirRqk1rJjadDBjYjiANb6VGDha
Z3WX4Vu7O1en8WPuznakN1gbrNcGy/42V2gpMNIs/pswmpw3gkWWE+M/zjj5rjkWj26aRRsE
bUj0Ok7umLm7g7TR7H2BtQjQ3GPXgE87ezrAjwn3iX6rXhnohhVgHXzF7KraaDeagcWNInZG
2sfrxY5IHFI6YwGx8Ba5z9ACK9r+UfadsW4IFljnK0KlpoQifwks7T/OE6p19SA+zGEB7xAv
zxKxOxOBiZcXuPjIRYYxt0hytRJPgAXaO9IDDKzAXpjwV1EcAawY0DDABsUnk7xCXZDVcA4s
uWFeliycKl3DKRcllU4lWWY41gIsxVfgpWqBxXSWRC1yOAVpcw6sTpSMjnBBsvh+3IgUdASf
6XK+oOBy0bcYWIBwuRteWIaHHh0VkkQrW3MSLGbuxchF4Oe55QkMvqUbtxDc5WX4gFPzrr7R
dqR/jqLZYO3HBmsdlw2WMi6bdhY6lWSDhuSXSOPEDX4IvaLP5CEUbGANqN/MobBg6tAXMbLJ
6SkzDRaGnanTjlQXMVGF404syq8Mgfh24zdjBHsFrGisGnPjedIX+QxJNHM0kscXM9W1ANZX
os4JsMbEAAFWnIV50+nMFBnxGNCfmUTM8eReghVphJjV1AGLe58KLCR+HdMBbBksHC4jbefF
Pidnucg2wNSG0Y/6yl6TZ5eZLQ2wSK4DPttZsEg2Ykey4mx7YPmQM2JMYxD9wMmJCFbGHaLL
b5clKwOrChAQhkaweaChK1rBPSUkQPHzxRde2TJYrsFKFPwCWPPXm2BZteGyMY9akBCKDsrP
FhV8QhAlpoN7ORNhcHTBsjZYLP1bU4lMsXfAQoXaNUqHS2Ni3FNSIISLpFEaB0ZzGGMkBTOa
nab2glGqSK93c3dehhh6D7Ds5848RZC207wpmmfBsg3bBuuZZYgZ7Q1Va8m3wktjBvMPKxET
LrVxQmslEMoQrfNCmA34IiE1XSciZOUqWG7d8b8PWAqgRvFEYjHPBvzMDcCyI27aWDkYKOPh
N/IzmHuUE6SiJi8Dy6qINHVDnCY9kApCkjkJHMvkNlnFiwJqKpmnYkrxgJVWnvUMr91xRtFA
ITdLp+F5ftEVi0lYPbAwbbEGi0RSOkWuXNeyZIkaLOarfwdYnE+hYDG+VKZdviJYlPZbB0vX
1PFac1NRm4fA4rNZBotkWS6DxasYCZtkUme1s9qKgvJxX8O9z/qsU6jcZW8hWPyk4lzuccfM
+Czab8OSOM+jjnSWsLluELXDU99ql902itcH6/JxbhtHVgN1Biuovr+Bxtt81h8m/5oRvBcE
K/brwcVvueepJi0ahvAys3rQoiyMWkzMdDBqMeAUsS+HzAUgJhShfUxazMSqMCsYJuGI6VOw
```

```
8HNVohDK3ky0jhjjhq6oqayHkKW1K0K2aM1FZy/TmWYyUwagDjHKsfsSsV5pHjxtPXHXdrii
qngvptMWSLIMEd8saotgwVoxzXA5rKfCBEdEpkUn6BWHxMDUyk8zzajkoKvhQzJHEyxa9ZIR
sdkFLHyxJNdhaOgTE96KRM02WJhcR0oZK7B0TbZJypQ22yqZSTpBg5y31vwvgBVUPiE7gs7S
CTLpOSmfo/2WsgFYzCX0p8Ca9wSZqMR3w5Ng6druHliephy5PwCW44URXae8tLOKEYhkvFt1
1hSnuB8siJv0wYKIA6Wx1Av33m7IX7ApjQyZjXmnhQ7FEepuGFhqPu7/Umdd9sZeuTfP+ZEF
+9fupDB+G1bRQ/ntFM2tMrD5rKfBEkkm94/sI7H76HWq+6lkTn/tiJO3ihK6hBRix8Z9ras6
ek1Wjn8ff74azL4CWA2sUKC6Z6nN1LeSrBZWrApjESyb6SD96ZeWrNX9lJiDq4zS2+osFqU1
Uo0ythZfs2qokxQmOQx2QbKAoeBUtEEZ2lAngm/lkpXUpUVqgeSTYSS8T9HQq36XZLFazjQ/
sidZfbC0rccpcFH5R4Z/4FFLFvcRUWoSyXJOvamuiqS5UwcspaOSjnKKun5GslBGkvLyrrN0
FqysnRlr2hkyaZDev6CzfiNYB0Hsbm5lydrrSpabGakTjCRRvfcX7IVqc3W7nfUoWN/Nljxt
Z70/WNbvlbgl6xtZhz8F1otL1lIfys4oBZFd/GbFy8tbrhf9Pg3a+sN/LHxmg7XB2mBtsP4W
WEf41mLDI4sNsqYPhwf9DJtSqGkU90QlX39Nyarp9YbBsiZZ8pZrj0kWf3TeYg24pPh0JCt5
aPLR3KuvpjNFsBY4+I6saRXTorobctTRYh2jlHrzG6wN1gZrg7XB2mBtsDZYPwLWGz2+J8Xz
14NVBJvGdhnr1fcm67IuhCmY87Uwp/rY2fosnVRsOtQES5UbdeHDe3tXhAp2Pa+lR7vvV8Aq
V3MNFuk/zj5N2lDojjkaK19q3NTgOloxX2fNxzVYhr07PBkRtvAtiybiPuSkeobGusUt8FIc
mynCpL1WA6wxBZL0AuJCw2pcst4vE6039SIy0rgJ1rV3Ou+E6SyChf3VJFjuuWTxNa7Ampqs
uZMWRKGQsFyGtWRZU+dVwrcAFtYuXpCsFKxYppgvwzpfL0VCphs+Ahaa0vTmiUlGs7otaApW
vhs6y3tsLcNrYFkOVriHi+fqI6kRL1Q33kK0vfWvmfLZTVqv7oZzDaAAi93/2sUtBufqgdS4
Mx7JaZd33GMJXzLRf7zo7psH8N6+4QZrg7XB+muPDdYGa4O1wdpg/dLHfwIMAORIgm4Mk35I
AAAAAElFTkSuQmCC
```

Figure 1: Code Page 437 Grapheme Reference

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.

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

NU$	0u	0U	cH-	cD-	cC	cS	BL$	BS$	HT$	LF$	Ml	Fm	CR$	M2	SU
$>	$<	UD	!*2	PI	SE	SR	$B	-!	-v	$$	EC$	-L	<>	UT	Dt
SP	!	"	Nb	DO	%	&	’	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
At	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	<(	//	)>	’>	_
’!	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	(!	!!	!)	’?	Eh
C,	u:	e’	a>	a:	a!	aa	c,	e>	e:	e!	i:	i>	i!	A:	AA
E’	ae	AE	o>	o:	o!	u>	u!	y:	O:	U:	Ct	Pd	Ye	Pt	Fl
a’	i’	o’	u’	n?	N?	-a	-o	?I	NI	NO	12	14	!I	<<	>>
.S	:S	?S	vv	vl	vL	Vl	Dl	dL	VL	VV	LD	UL	Ul	uL	dl
ur	uh	dh	vr	hh	vh	vR	Vr	UR	DR	UH	DH	VR	HH	VH	uH
Uh	dH	Dh	Ur	uR	dR	Dr	Vh	vH	ul	dr	FB	LB	lB	RB	TB
a*	$$	G*	$$	$$	s*	$$	t*	F*	H*	$$	$$	00	$$	$$	(U
=3	+-	>=	=<	Iu	Il	-:	?2	Ob	.M	Sb	RT	nS	2S	fS	NS$

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.

```
MIME-Version: 1.0
Content-Type: text/nfo
Content-Disposition: attachment; filename="RELEASE.NFO"
Content-Transfer-Encoding: base64

TODO/PutInBase64EncodedContentHere==
```

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

[ANSI.SYS] Microsoft Corporation, "ANSI.SYS", MSDN ID cc722862, 1994, <http://technet.microsoft.com/library/cc722862>.

[OEMCP437] Microsoft Corporation, "OEM 437", MSDN ID cc305156, 2014, <http://msdn.microsoft.com/goglobal/cc305156>.

[RFC1345] Simonsen, K., "Character Mnemonics and Character Sets", RFC 1345, June 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5147] Wilde, E. and M. Duerst, "URI Fragment Identifiers for the text/plain Media Type", RFC 5147, April 2008.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[UTF] The Unicode Consortium, "The Unicode Standard, Version 8.0.0", Chapter 3: "Conformance", The Unicode Consortium, August 2015.

7.2. Informative References

[ISO2022] International Organization for Standardization, "Character Code Structure and Extension Techniques, 6th edition", ISO Standard 2022, ECMA-35, December 1994.

[ISO6429] International Organization for Standardization, "Information Technology - Control Functions for Coded Character Sets, 3rd edition", ISO Standard 6429, December 1992.

[ISO8613] International Organization for Standardization, "Information Technology - Open Document Architecture (ODA) and Interchange Format: Character Content Architectures", ISO Standard 8613-6, ITU-T T.416, March 1993.

[REG196] International Organization for Standardization, "International Register of Coded Character Sets: UTF-8 without implementation level", Sec. 2.8.1, Reg. 196, April 1996, <http://kikaku.itscj.ipsj.or.jp/ISO-IR/196.pdf>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[SAUCE] O. "Tasmaniac" Reubens / ACiD, "SAUCE--Standard Architecture for Universal Comment Extensions", 00.5, November 2013, <http://www.acid.org/info/sauce/sauce.htm>.

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 Blvd
Ste 2600
Los Angeles, CA 90036
USA

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