PharmaSUG China 2024 - Paper AP-10080

Fighting with Special Characters in Data Management Programmer's Daily Work

Fei Guo, Sanofi

ABSTRACT

The presence of special characters in clinical trial data can significantly hinder the generation of high-quality deliverables. This can manifest in several ways, ranging from major discrepancies like inaccurate statistics and participant counts to minor inconveniences such as misplaced line breaks, unintended page breaks, and the cryptic symbols within reports.

In the pharmaceutical and biotech industries, where data integrity is paramount, addressing these character issues becomes an essential step. To achieve this, a thorough understanding of the root causes is crucial. By employing a combination of effective techniques and some communication methodology, we can successfully pinpoint and rectify these problems, ensuring the accuracy and professionalism of our final deliverables.

INTRODUCTION

Identifying and eliminating special characters is an important task for DMP, it is not only a technical approach but also a transversal task which need communication skill. In this article, we will show you the whole picture of special character and the methodology regarding how to fetch them, then you will be immersed in a daily work scenario of DMP, where we will show you the complete process from discovering special characters to completely solving the problems caused by them.

BACKGROUND INFORMATION

Data Management Programmer (DMP) is also known as SAS programmer in some pharma companies, a typical distinction between DMP and Biostatistical Programmer (BP, or StatP) is the different backgrounds of their upstream role/customers. The major upstream role of DMP is Data Manager (DM) or medical (e.g., Study Medical Manager, Clinical Scientist) / clinical operational people (e.g., Global or Local Study Manager, Site Monitor). For major upstream role of BP is usually DMP or DM who have sufficient programming skills to provide datasets which is used for further derivation by BP (e.g., SDTM and ADaM dataset).

In most cases, if the DMP deliverables contain special characters, it should come from the source files provided by upstream role. A typical scenario is, Study Medical Manager copy some information contains special character (e.g., age ⩾ 18) from protocol into 'list of predefined protocol deviations' which will be used to generate detected protocol deviation output by DMP, then ⩾ is flowed into the raw data used by BP to create SDTM and ADaM datasets, which may have a negative impact when submit to agency.

DEFINITION OF SPECIAL CHARACTER

'Special character' in computer science is a relative concept which may have different definition depends on which character encoding standards, software, and character encodings (i.e., character sets) you used. We are talking about DMP's daily work in this article, so we will base on below assumption:

Character encoding standards: The Unicode (i.e., UCS, Universal Coded Character Set) Standard, ASCII (American Standard Code for Information Interchange)

Software: SAS

Encoding: LATIN1 (i.e., Latin1; ISO-8859-1)

ASCII

ASCII (i.e., Standard ASCII) is a character encoding standard for electronic communication, which is a 7-bit character code where each individual code which contains 7 bits (number 0 or 1) represents a unique character from early in its development. It is also known as lower ASCII which contains decimal value of 0-127 (2^7=128).

As computer technology spread throughout the world, different standards bodies and corporations developed many variations of ASCII to facilitate the expression of non-English languages, which is considered as extended ASCII table with 8-bit (1 byte) character code including 256 characters and symbols (2^8=256), it is also known as upper ASCII.

Please aware that there is no formal definition of extended ASCII, and even use of the term is sometimes criticized, because it can be mistakenly interpreted. So, when you hear someone talk about ASCII, it would be great to further confirm if it means standard ASCII (0-127) or extended ASCII (0-255) to avoid misunderstanding.

Extended ASCII is a repertoire of character encodings that include 95 printable characters (decimal value 32-126) which means they can be displayed or printed on paper (including lowercase/uppercase letters, digits and some most common used punctuation marks and symbols, you can find almost every character on your keyboard), plus 161 additional characters (0-31 and 127 which is called control characters, as well as 128-255) which is considered as non-printable special characters, i.e., NPSC. Extended ASCII encodings are platform- and locale-dependent, there are different implementations of the ASCII table, for example:

Extended ASCII Table for ISO-8859-1: particularly designed to support Latin1/Western European languages (WLATIN1).

Extended ASCII Table for ISO-8859-2: particularly designed to support Latin2/Eastern European languages.

Extended ASCII Table of Windows-1252 (CP-1252): using displayable characters rather than control characters in the 128 to 159 range, it is probably the most used 8-bit character encoding in the world.

Extended ASCII table for code page 437: particularly designed to support Western Europe languages.

Among the different variations of the ASCII table, the decimal value 32-127 are common for all, the only difference is in the 128-255.

Table 1-3 shows the extended ASCII table for ISO-8859-1.

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
0	000	00	00000000	NUL	&#00;		Null character
1	001	01	00000001	SOH	&#01;		Start of Heading
2	002	02	00000010	STX	&#02;		Start of Text
3	003	03	00000011	ETX	&#03;		End of Text
4	004	04	00000100	EOT	&#04;		End of Transmission
5	005	05	00000101	ENQ	&#05;		Enquiry
6	006	06	00000110	ACK	&#06;		Acknowledge
7	007	07	00000111	BEL	&#07;		Bell, Alert
8	010	08	00001000	BS	&#08;		Backspace
9	011	09	00001001	HT			Horizontal Tab
10	012	0A	00001010	LF
	Line Feed
11	013	0B	00001011	VT	&#11;		Vertical Tabulation
12	014	0C	00001100	FF			Form Feed
13	015	0D	00001101	CR
	Carriage Return
14	016	0E	00001110	SO	&#14;		Shift Out
15	017	0F	00001111	SI	&#15;		Shift In
16	020	10	00010000	DLE	&#16;		Data Link Escape

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
17	021	11	00010001	␑	&#17;		Device Control One (XON)
18	022	12	00010010	␒	&#18;		Device Control Two
19	023	13	00010011	␓	&#19;		Device Control Three (XOFF)
20	024	14	00010100	␔	&#20;		Device Control Four
21	025	15	00010101	␕	&#21;		Negative Acknowledge
22	026	16	00010110	␖	&#22;		Synchronous Idle
23	027	17	00010111	␗	&#23;		End of Transmission Block
24	030	18	00011000	␘	&#24;		Cancel
25	031	19	00011001	␙	&#25;		End of medium
26	032	1A	00011010	␚	&#26;		Substitute
27	033	1B	00011011	␛	&#27;		Escape
28	034	1C	00011100	␜	&#28;		File Separator
29	035	1D	00011101	␝	&#29;		Group Separator
30	036	1E	00011110	␞	&#30;		Record Separator
31	037	1F	00011111	␟	&#31;		Unit Separator

Table 1. ASCII control characters (character code 0-31): the first 32 characters in the ASCII-table are unprintable control codes and are used to control peripherals such as printers

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000	␠			Space
33	041	21	00100001	!	!	!	Exclamation mark
34	042	22	00100010	"	"	"	Double quotes (or speech marks)
35	043	23	00100011	#	#	#	Number sign
36	044	24	00100100	$	$	$	Dollar
37	045	25	00100101	%	%	%	Per cent sign
38	046	26	00100110	&	&	&	Ampersand
39	047	27	00100111	'	'	'	Single quote
40	050	28	00101000	(	(	&lparen;	Open parenthesis (or open bracket)
41	051	29	00101001	)	)	&rparen;	Close parenthesis (or close bracket)
42	052	2A	00101010	*	*	*	Asterisk
43	053	2B	00101011	+	+	+	Plus
44	054	2C	00101100	,	,	,	Comma
45	055	2D	00101101	-	-		Hyphen-minus
46	056	2E	00101110	.	.	.	Period, dot or full stop
47	057	2F	00101111	/	/	/	Slash or divide
48	060	30	00110000	0	0		Zero
49	061	31	00110001	1	1		One
50	062	32	00110010	2	2		Two
51	063	33	00110011	3	3		Three
52	064	34	00110100	4	4		Four

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
53	065	35	00110101	5	&#53;		Five
54	066	36	00110110	6	&#54;		Six
55	067	37	00110111	7	&#55;		Seven
56	070	38	00111000	8	&#56;		Eight
57	071	39	00111001	9	&#57;		Nine
58	072	3A	00111010	:	&#58;	&colon;	Colon
59	073	3B	00111011	;	&#59;	&semi;	Semicolon
60	074	3C	00111100	<	&#60;	&lt;	Less than (or open angled bracket)
61	075	3D	00111101	=	&#61;	&equals;	Equals
62	076	3E	00111110	>	&#62;	&gt;	Greater than (or close angled bracket)
63	077	3F	00111111	?	&#63;	&quest;	Question mark
64	100	40	01000000	@	&#64;	&commat;	At sign
65	101	41	01000001	A	&#65;		Uppercase A
66	102	42	01000010	B	&#66;		Uppercase B
67	103	43	01000011	C	&#67;		Uppercase C
68	104	44	01000100	D	&#68;		Uppercase D
69	105	45	01000101	E	&#69;		Uppercase E
70	106	46	01000110	F	&#70;		Uppercase F
71	107	47	01000111	G	&#71;		Uppercase G
72	110	48	01001000	H	&#72;		Uppercase H
73	111	49	01001001	I	&#73;		Uppercase I
74	112	4A	01001010	J	&#74;		Uppercase J
75	113	4B	01001011	K	&#75;		Uppercase K
76	114	4C	01001100	L	&#76;		Uppercase L
77	115	4D	01001101	M	&#77;		Uppercase M
78	116	4E	01001110	N	&#78;		Uppercase N
79	117	4F	01001111	O	&#79;		Uppercase O
80	120	50	01010000	P	&#80;		Uppercase P
81	121	51	01010001	Q	&#81;		Uppercase Q
82	122	52	01010010	R	&#82;		Uppercase R
83	123	53	01010011	S	&#83;		Uppercase S
84	124	54	01010100	T	&#84;		Uppercase T
85	125	55	01010101	U	&#85;		Uppercase U
86	126	56	01010110	V	&#86;		Uppercase V
87	127	57	01010111	W	&#87;		Uppercase W
88	130	58	01011000	X	&#88;		Uppercase X
89	131	59	01011001	Y	&#89;		Uppercase Y
90	132	5A	01011010	Z	&#90;		Uppercase Z
91	133	5B	01011011	[	&#91;	&lsqb;	Opening bracket
92	134	5C	01011100	\	&#92;	&bsol;	Backslash
93	135	5D	01011101	]	&#93;	&rsqb;	Closing bracket
94	136	5E	01011110	^	&#94;	&Hat;	Caret - circumflex
95	137	5F	01011111	_	&#95;	&lowbar;	Underscore
96	140	60	01100000	`	&#96;	&grave;	Grave accent
97	141	61	01100001	a	&#97;		Lowercase a

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
98	142	62	01100010	b	&#98;		Lowercase b
99	143	63	01100011	c	&#99;		Lowercase c
100	144	64	01100100	d	&#100;		Lowercase d
101	145	65	01100101	e	&#101;		Lowercase e
102	146	66	01100110	f	&#102;		Lowercase f
103	147	67	01100111	g	&#103;		Lowercase g
104	150	68	01101000	h	&#104;		Lowercase h
105	151	69	01101001	i	&#105;		Lowercase i
106	152	6A	01101010	j	&#106;		Lowercase j
107	153	6B	01101011	k	&#107;		Lowercase k
108	154	6C	01101100	l	&#108;		Lowercase l
109	155	6D	01101101	m	&#109;		Lowercase m
110	156	6E	01101110	n	&#110;		Lowercase n
111	157	6F	01101111	o	&#111;		Lowercase o
112	160	70	01110000	p	&#112;		Lowercase p
113	161	71	01110001	q	&#113;		Lowercase q
114	162	72	01110010	r	&#114;		Lowercase r
115	163	73	01110011	s	&#115;		Lowercase s
116	164	74	01110100	t	&#116;		Lowercase t
117	165	75	01110101	u	&#117;		Lowercase u
118	166	76	01110110	v	&#118;		Lowercase v
119	167	77	01110111	w	&#119;		Lowercase w
120	170	78	01111000	x	&#120;		Lowercase x
121	171	79	01111001	y	&#121;		Lowercase y
122	172	7A	01111010	z	&#122;		Lowercase z
123	173	7B	01111011	{	&#123;	&lcub;	Opening brace
124	174	7C	01111100 |		&#124;	&verbar;	Vertical bar
125	175	7D	01111101	}	&#125;	&rcub;	Closing brace
126	176	7E	01111110	~	&#126;	&tilde;	Equivalency sign - tilde
127	177	7F	01111111		&#127;		Delete

Table 2. ASCII printable characters (decimal value 32-126): Codes 32-127 are common for all the different variations of the ASCII table (Note: character 127 represents the command DEL which should be considered as control character)

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
128	200	80	10000000				Unused
129	201	81	10000001				Unused
130	202	82	10000010				Unused
131	203	83	10000011				Unused
132	204	84	10000100				Unused
133	205	85	10000101				Unused
134	206	86	10000110				Unused
135	207	87	10000111				Unused
136	210	88	10001000				Unused
137	211	89	10001001				Unused

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
138	212	8A	10001010				Unused
139	213	8B	10001011				Unused
140	214	8C	10001100				Unused
141	215	8D	10001101				Unused
142	216	8E	10001110				Unused
143	217	8F	10001111				Unused
144	220	90	10010000				Unused
145	221	91	10010001				Unused
146	222	92	10010010				Unused
147	223	93	10010011				Unused
148	224	94	10010100				Unused
149	225	95	10010101				Unused
150	226	96	10010110				Unused
151	227	97	10010111				Unused
152	230	98	10011000				Unused
153	231	99	10011001				Unused
154	232	9A	10011010				Unused
155	233	9B	10011011				Unused
156	234	9C	10011100				Unused
157	235	9D	10011101				Unused
158	236	9E	10011110				Unused
159	237	9F	10011111				Unused
160	240	A0	10100000				Non-breaking space
161	241	A1	10100001	¡	&#161;	&iexcl;	Inverted exclamation mark
162	242	A2	10100010	¢	&#162;	&cent;	Cent sign
163	243	A3	10100011	£	&#163;	&pound;	Pound sign
164	244	A4	10100100	¤	&#164;	&curren;	Currency sign
165	245	A5	10100101	¥	&#165;	&yen;	Yen sign
166	246	A6	10100110	¦	&#166;	&brvbar;	Pipe, broken vertical bar
167	247	A7	10100111	§	&#167;	&sect;	Section sign
168	250	A8	10101000	¨	&#168;	&uml;	Spacing diaeresis - umlaut
169	251	A9	10101001	©	&#169;	&copy;	Copyright sign
170	252	AA	10101010	ª	&#170;	&ordf;	Feminine ordinal indicator
171	253	AB	10101011	«	&#171;	&laquo;	Left double angle quotes
172	254	AC	10101100	¬	&#172;	&not;	Negation
173	255	AD	10101101		&#173;	&shy;	Soft hyphen
174	256	AE	10101110	®	&#174;	&reg;	Registered trade mark sign
175	257	AF	10101111	¯	&#175;	&macr;	Spacing macron - overline
176	260	B0	10110000	°	&#176;	&deg;	Degree sign
177	261	B1	10110001	±	&#177;	&plusmn;	Plus-or-minus sign
178	262	B2	10110010	²	&#178;	&sup2;	Superscript two - squared

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
179	263	B3	10110011	³	³	³	Superscript three - cubed
180	264	B4	10110100	´	´	´	Acute accent - spacing acute
181	265	B5	10110101	µ	µ	µ	Micro sign
182	266	B6	10110110	¶	¶	¶	Pilcrow sign - paragraph sign
183	267	B7	10110111	·	·	·	Middle dot - Georgian comma
184	270	B8	10111000	¸	¸	¸	Spacing cedilla
185	271	B9	10111001	¹	¹	¹	Superscript one
186	272	BA	10111010	º	º	º	Masculine ordinal indicator
187	273	BB	10111011	»	»	»	Right double angle quotes
188	274	BC	10111100	¼	¼	¼	Fraction one quarter
189	275	BD	10111101	½	½	½	Fraction one half
190	276	BE	10111110	¾	¾	¾	Fraction three quarters
191	277	BF	10111111	¿	¿	¿	Inverted question mark
192	300	C0	11000000	À	À	À	Latin capital letter A with grave
193	301	C1	11000001	Á	Á	Á	Latin capital letter A with acute
194	302	C2	11000010	Â	Â	Â	Latin capital letter A with circumflex
195	303	C3	11000011	Ã	Ã	Ã	Latin capital letter A with tilde
196	304	C4	11000100	Ä	Ä	Ä	Latin capital letter A with diaeresis
197	305	C5	11000101	Å	Å	Å	Latin capital letter A with ring above
198	306	C6	11000110	Æ	Æ	Æ	Latin capital letter AE
199	307	C7	11000111	Ç	Ç	Ç	Latin capital letter C with cedilla
200	310	C8	11001000	È	È	È	Latin capital letter E with grave
201	311	C9	11001001	É	É	É	Latin capital letter E with acute
202	312	CA	11001010	Ê	Ê	Ê	Latin capital letter E with circumflex
203	313	CB	11001011	Ë	Ë	Ë	Latin capital letter E with diaeresis
204	314	CC	11001100	Ì	Ì	Ì	Latin capital letter I with grave
205	315	CD	11001101	Í	Í	Í	Latin capital letter I with acute
206	316	CE	11001110	Î	Î	Î	Latin capital letter I with circumflex
207	317	CF	11001111	Ï	Ï	Ï	Latin capital letter I with diaeresis
208	320	D0	11010000	Ð	Ð	Ð	Latin capital letter ETH

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
209	321	D1	11010001	Ñ	Ñ	Ñ	Latin capital letter N with tilde
210	322	D2	11010010	Ò	Ò	Ò	Latin capital letter O with grave
211	323	D3	11010011	Ó	Ó	Ó	Latin capital letter O with acute
212	324	D4	11010100	Ô	Ô	Ô	Latin capital letter O with circumflex
213	325	D5	11010101	Õ	Õ	Õ	Latin capital letter O with tilde
214	326	D6	11010110	Ö	Ö	Ö	Latin capital letter O with diaeresis
215	327	D7	11010111	×	×	×	Multiplication sign
216	330	D8	11011000	Ø	Ø	Ø	Latin capital letter O with slash
217	331	D9	11011001	Ù	Ù	Ù	Latin capital letter U with grave
218	332	DA	11011010	Ú	Ú	Ú	Latin capital letter U with acute
219	333	DB	11011011	Û	Û	Û	Latin capital letter U with circumflex
220	334	DC	11011100	Ü	Ü	Ü	Latin capital letter U with diaeresis
221	335	DD	11011101	Ý	Ý	Ý	Latin capital letter Y with acute
222	336	DE	11011110	Þ	Þ	Þ	Latin capital letter THORN
223	337	DF	11011111	ß	ß	ß	Latin small letter sharp s - ess-zed
224	340	E0	11100000	à	à	à	Latin small letter a with grave
225	341	E1	11100001	á	á	á	Latin small letter a with acute
226	342	E2	11100010	â	â	â	Latin small letter a with circumflex
227	343	E3	11100011	ã	ã	ã	Latin small letter a with tilde
228	344	E4	11100100	ä	ä	ä	Latin small letter a with diaeresis
229	345	E5	11100101	å	å	å	Latin small letter a with ring above
230	346	E6	11100110	æ	æ	æ	Latin small letter ae
231	347	E7	11100111	ç	ç	ç	Latin small letter c with cedilla
232	350	E8	11101000	è	è	è	Latin small letter e with grave
233	351	E9	11101001	é	é	é	Latin small letter e with acute
234	352	EA	11101010	ê	ê	ê	Latin small letter e with circumflex
235	353	EB	11101011	ë	ë	ë	Latin small letter e with diaeresis

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
236	354	EC	11101100	ì	ì	ì	Latin small letter i with grave
237	355	ED	11101101	í	í	í	Latin small letter i with acute
238	356	EE	11101110	î	î	î	Latin small letter i with circumflex
239	357	EF	11101111	ï	ï	ï	Latin small letter i with diaeresis
240	360	F0	11110000	ð	ð	ð	Latin small letter eth
241	361	F1	11110001	ñ	ñ	ñ	Latin small letter n with tilde
242	362	F2	11110010	ò	ò	ò	Latin small letter o with grave
243	363	F3	11110011	ó	ó	ó	Latin small letter o with acute
244	364	F4	11110100	ô	ô	ô	Latin small letter o with circumflex
245	365	F5	11110101	õ	õ	õ	Latin small letter o with tilde
246	366	F6	11110110	ö	ö	ö	Latin small letter o with diaeresis
247	367	F7	11110111	÷	÷	÷	Division sign
248	370	F8	11111000	ø	ø	ø	Latin small letter o with slash
249	371	F9	11111001	ù	ù	ù	Latin small letter u with grave
250	372	FA	11111010	ú	ú	ú	Latin small letter u with acute
251	373	FB	11111011	û	û	û	Latin small letter u with circumflex
252	374	FC	11111100	ü	ü	ü	Latin small letter u with diaeresis
253	375	FD	11111101	ý	ý	ý	Latin small letter y with acute
254	376	FE	11111110	þ	þ	þ	Latin small letter thorn
255	377	FF	11111111	ÿ	ÿ	ÿ	Latin small letter y with diaeresis

Table 3. The extended ASCII codes (character code 128-255)

UNICODE

Unicode (defined by the international standard ISO/IEC 10646) is a text encoding standard maintained by the Unicode Consortium designed to support the use of text written in all the world's major writing systems, it has a much wider array of characters and their various encoding forms have begun to supplant previous standard ISO/IEC 8859 and ASCII rapidly in many environments. While ASCII is limited to 128 characters, Unicode and the UCS support more characters by separating the concepts of unique identification (using natural numbers called code points) and encoding (Unicode defines three major encodings: UTF-8, UTF-16, and UTF-32).

A single-byte character set (i.e., SBCS) is an encoding where each character is encoded with one byte. ASCII and extended ASCII encoding are SBCS encodings. When more than one byte is needed to represent a character, like in the UTF-8 (i.e., Unicode transformation format – 8-bit) encoding, the

character set is called a multibyte character set (MBCS), because UTF-8 can encode all 1,112,064 valid Unicode code points using one to four one-byte (8-bit) code units, depending on the value of the code point.

ASCII was incorporated into the Unicode (1991) character set as the first 128 symbols, so the ASCII characters have the same numeric codes in both sets. This allows UTF-8 to be backward compatible with 7-bit ASCII, as a UTF-8 file containing only ASCII characters is identical to an ASCII file containing the same sequence of characters. Even more importantly, forward compatibility is ensured as software that recognizes only 7-bit ASCII characters as special and does not alter bytes with the highest bit set (as is often done to support 8-bit ASCII extensions) will preserve UTF-8 data unchanged.

As a summary, ASCII is primarily designed for English and some basic European characters, whereas Unicode is a more comprehensive solution designed to achieve a unified encoding for the diverse scripts of the world, facilitating cross-lingual information processing and exchange. With the advancement of globalization and digitization, Unicode has become the preferred standard for character encoding in modern information systems, UTF-8 is by far the most dominant encoding on the world wide web.

ENCODING

In the clinical trial industry, especially in statistical data analysis and the submission of clinical trial data, SAS stands as one of the predominant software. Agencies such as the U.S. Food and Drug Administration (FDA), European Medicines Agency (EMA), China's National Medical Products Administration (NMPA) and Japan's Pharmaceuticals and Medical Devices Agency (PMDA), SAS widely adopted in submission practice although it may not be enforced to use. The prevalence of MRCT (Multi-regional Clinical Trial) and the necessity for data interoperability and comparability, making SAS files a common and highly compatible format for cross-border clinical research and pharmaceutical registration submissions.

SAS is a powerful software which allows user to define how a computer interprets and represents the values within data, which is called encoding. The SAS session encoding is set using the ENCODING system option, it is a startup option only. You can use one of the following methods to check your session encoding:

```
proc options option=encoding; run;

%put SESSION ENCODING is &sysencoding;

%put SESSION Encoding=%sysfunc(getoption(encoding));
```

This will generate a log message showing what encoding your SAS session is using.

```
ENCODING=LATIN1   Specifies the default character-set encoding for the SAS
session.
```

Output 1. Log message generated by PROC OPTIONS

When you need to change the default SAS session encoding (change LATIN1 into UTF-9 for example), following below the steps:

1. Edit SAS Configuration File: You need to edit the SAS configuration file to set the session encoding to UTF-8. The configuration file is typically named ‘sasv9.cfg’ (for SAS 9.x versions) or ‘sasviya.cfg’ (for Viya environments), and it's located in the SAS installation directory, usually under SASHome\SASFoundation\9.5 (or the respective version number) or SASHome\Lev1\SASConfig_\Viya.
2. Edit the Configuration File:
    1) Open the ‘sasv9.cfg’ or ‘sasviya.cfg’ file with a text editor (like Notepad++, or any other plain text editor).
    2) Add the following line at the end of the file to set the session encoding to UTF-8:

```
-encoding utf-8
```

3. Save and Restart SAS: Save the changes to the configuration file and then restart your SAS session for the changes to take effect. If you're using SAS Enterprise Guide, SAS Studio, or another SAS interface, close and reopen the application.

This option tells SAS to use UTF-8 as the default character encoding for the session. Note that this action also changes the encoding used in every SAS configured component. These components include but are not limited to the following: workspace servers, stored process servers, and OLAP Servers.

If you don't have administrative access to modify the global configuration file, and you're using SAS interactively (like through SAS Display Manager or SAS Studio), you can also set the encoding on a per-session basis by using the -encoding option when starting SAS from the command line. However, this approach is less practical for routine use than modifying the configuration file.

Unfortunately, administrative access is prohibited in most companies per safety and compliance consideration. Now, let's go back to our previous assumption, you should know why we indicate the scenario of using SAS with LATIN1 encoding, here are a few reasons why LATIN1 may have been the default encoding session:

1. Legacy compatibility: SAS has been around since the 1970s, well before Unicode and UTF-8 became prevalent standards. Many older systems and applications were built with LATIN1 or similar single-byte encodings in mind.

2. Resource utilization: LATIN1 uses one byte per character, which is more memory-efficient compared to multi-byte encodings like UTF-8. In the past, when computational resources were more limited, this could have been a significant consideration.

3. Widespread use: LATIN1 was a standard across many platforms and applications, ensuring compatibility with a broad range of systems without requiring complex encoding conversions.

4. Agency requirement: take FDA for example, per the data submission guidelines, "Variable names, as well as variable and dataset labels should include American Standard Code for Information Interchange (ASCII) text codes only. Variable values are the most broadly compatible with software and operating systems when they are restricted to ASCII text codes (printable values below 128). Use UTF-8 for extending character sets; however, the use of extended mappings is not recommended. Transcoding errors, variable length errors, and lack of software support for multi byte UTF-8 encodings can result in incorrect character display and variable value truncations."

Based on above assumption, we can give a definition of special characters: characters not in ASCII printable characters (decimal value 32-126) listed in table 2.

HARMFULNESS OF SPECIAL CHARACTER

If the imported data contains special characters, what will happened?

The encoding of a data set or external file being read may differ from the session encoding. In that case, the data set must be transcoded into the session encoding for SAS to work with it. For SAS data sets, transcoding is usually done automatically with CEDA (Cross Environment Data Access) which allows SAS to do the work behind the scenes. Assuming no issues are encountered, you are left with a simple note in the log:

```
NOTE: Data file PREDEFINE.DATA is in a format that is native to another host,
or the file encoding does not match the session encoding. Cross Environment
Data Access will be used, which might require additional CPU resources and
might reduce performance.
```

Unfortunately, in many cases you might not be the lucky guy. If SAS detects an error during transcoding, a message like below is displayed in the log without explicit explanation and location of the issue:

```
ERROR: Some character data was lost during transcoding in the dataset
PREDEFINE. Either the data contains characters that are not representable in
the new encoding or truncation occurred during transcoding.
```

The error message suggests two possible reasons:

1. Characters that are not representable means that the data contains characters that do not exist in the session encoding which support a smaller set of characters compared to source encoding.

2. Truncation means that a variable in the new data set does not have enough room to receive the transcoded data and therefore truncation occurred.

For example: The French character é (e with acute accent) is made up on one byte in WLATIN1 (binary value: 11101001), in UTF-8 it is made up of 2 bytes (binary representation: 11000011 10101001). If we have a variable value equal to “idée”, and DMP define the variable length is $4, the second byte (10101001) of é in string “idée” will be truncated. Then é will be interpreted as Ã (Latin capital letter A with tilde) in ASCII (binary value: 11000011), as a result, “idée” will be changed into “idÃe” falsely.

Another example is: the Greek letter beta (β which belongs to the Unicode character set), are incorrectly attempted to be imported into SAS using LATIN1 encoding, SAS may recognize them as substitute characters or produce garbled text because these characters do not have a corresponding code in the LATIN1 encoding. Particularly, if the beta character is mishandled as a single byte encoding and its byte value falls within the range of ASCII control characters, it could theoretically be interpreted as the substitute character in ASCII (decimal value 26). Figure 1-3 shows the β displayed in different files:

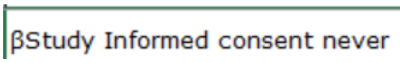


Figure 1. Imported excel file contains β

Figure 2. SAS interpret β as the substitute character

Figure 3. Excel output file exported by SAS

In a summary, special characters are harmful on DMP deliverables, also impact downstream deliverables such as SDTM and ADaM datasets, including:

1. Data Import/Export Issues: As mentioned, non-ASCII or non-LATIN1 characters, if not properly encoded, can lead to garbled text, or data loss. This directly affects the readability and accuracy of the data.

2. Analytical Errors: If special characters are incorrectly encoded or replaced in critical variables (like laboratory results, adverse event descriptions, etc.), it can impair the accuracy of data analysis. For instance, a drug name containing a special character, if mishandled, might lead to incorrect categorization and analysis of the drug-related event.

3. Data Consistency Issues: Inconsistency in handling the same special character across different records within a dataset (where some are correctly encoded and others are not) is a violation of data consistency, making data comparisons becomes more complex or even unfeasible.

4. Compliance with Standards: Pharma agency and clinical standards like CDISC require that data submitted to regulatory bodies adhere to specific formatting and encoding rules. Mismanagement of special characters can violate these guidelines, resulting in data submissions being rejected or necessitating resubmission.

5. System Compatibility Issues: Different database systems, statistical software, and Electronic Data Capture System (EDC) vary in their support for character encoding. Incorrect handling of special characters can create obstacles when transferring data between systems, impeding smooth data exchange.

FIGHTING WITH SPECIAL CHARACTERS

Based on above conclusions, DMP must identify and deal with special characters to make sure their deliverables will not be impacted.

IDENTIFICATION OF SPECIAL CHARACTERS

Considering of efficiency and accuracy, it is necessary to develop a tool to identifying and highlighting characters that may cause problems automatically. Now let's look at how a DMP should handle special characters in its daily work from the perspective of a real working scenario.

Checking points

DMP Ping has been assigned to a protocol deviation (PD) programming task, the upstream user (DM David) send Ping a 'List of predefined potential deviations' excel list (PD list) which is drafted and approved by Study Medical Manager (SMM) Maiko, then Ping will draft a excel protocol deviation specification (PD spec) based on the PD list, then setup a series of SAS PD programs following the aligned programming condition in PD spec. Ping will also setup an excel PD matrix to define and control all the PD programs. In another hand, Study Monitor (SM) Singh will record all the manual PDs in a dedicated collection system during his monitoring visit, this PD is called manual PD which will flow into a data warehouse.

When all PD programs are passed UAT and go-live, Ping will run the program to read the matrix and extract manual PD dataset from the data warehouse, then generate excel PD outputs accordingly and send it (including outputs from all automatic PD programs as well as manual PD) to study team for review by monthly. During each study milestone (e.g., interim analysis, database lock), Ping will also generate two SAS PD outputs (SAS dataset, one is automatic PD, another one is manual PD) and send to BP Billy for further statistically analysis usage (derive SDTM and ADaM dataset for submission purpose). The assignment is shown in figure 4:

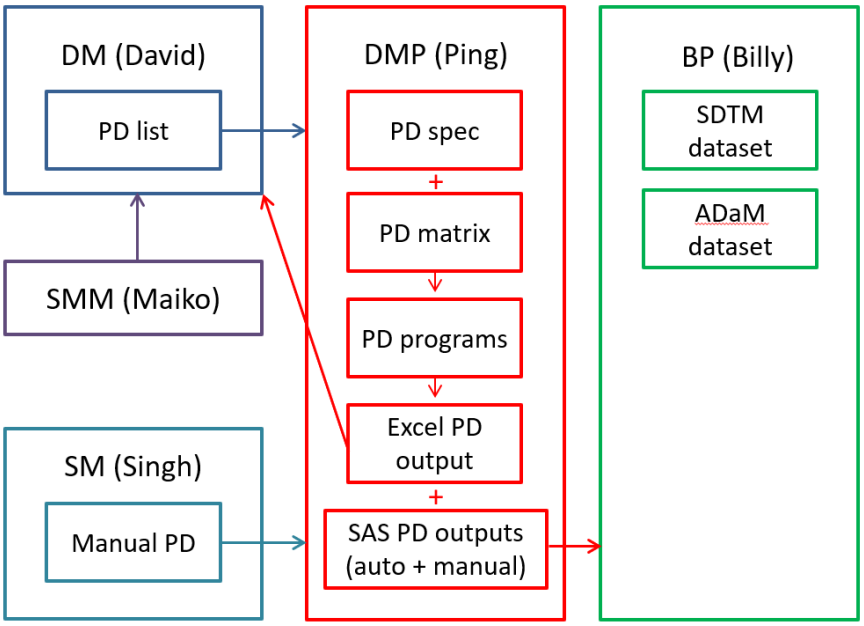


Figure 4. Protocol deviation (PD) programming task assignment

At the bottom line, Ping needs to make sure all the special characters are excluded in the final deliverable to Billy. So Ping developed a SAS macro called 'SAS_output_ASCII.sas' to generate a excel file to detect all the special characters in SAS PD output. Output 2 shows excel output:

Compound: NCR17375 Study: ACC17923							
Special characters in SAS PD datasets							
3 rows in this report							
Number	Folder	File_Name	Variable_Name	Obs	USUBJID	Original_Value	Special_Character
1	DEVOPS/NCR17375/ACC17923/CDM_DEVIAT/SDTM/DATA	ACC17923_bs_man.sas7bdat	DVCM	2	017924-604-00001	MELFALÁN	Á
2	DEVOPS/NCR17375/ACC17923/CDM_DEVIAT/SDTM/DATA	ACC17923_bs_aut.sas7bdat	DVDECOD	5	017924-604-00024	Concomitant Medications/<_carriage return_><_line feed_>Therapy	<_carriage return_><_line feed_>
3	DEVOPS/NCR17375/ACC17923/CDM_DEVIAT/SDTM/DATA	ACC17923_bs_man.sas7bdat	DVCM	7	017924-604-00008	≤18	≤

Output 2. Excel output generated by 'check_output_ASCII.sas'

Based on the output, Ping gets below information:

1. In manual PD dataset located in 'DEVOPS/NCR17375/ACC17923/CDM_DEVIAT/SDTM/DATA ACC17923_bs_man.sas7bdat', two special characters are found in DVCM variable, one is Á (decimal value 193) in observation 2, another one is '≤' (decimal value 243 in extended ASCII table for code page 437) in observation 7.

2. In automatic PD dataset located in 'DEVOPS/NCR17375/ACC17923/CDM_DEVIAT/SDTM/DATA ACC17923_bs_aut.sas7bdat', two special characters are found in DVDECOD variable: 'carriage return' (decimal value 13) + 'line feed' (decimal value 10) in observation 5 (caused by pressing 'Enter' in keyboard).

Next question is, the special character comes from which file? The answer is PD matrix because it's the only import file by SAS PD program. Furthermore, which file and who will impact the PD matrix content? Some of the content comes from the PD list copied by Ping, other contents are manually filled in by Ping (e.g., program name, program ID, program location, etc.)

Based on that, Ping decides to develop another SAS macro called 'PD_list_ASCII.sas' to generate a excel file to detect all the special characters in PD list. Output 3 shows excel output:

Compound: NCR17375 Study: ACC17923							
Special characters in PD list							
1 row in this report							
Number	Folder	File_Name	Variable_Name	Obs	USUBJID	Original_Value	Special_Character
1	DEVOPS/NCR17375/ACC17923/CDM_DEVIAT/SDTM/DATA	ACC17923_bs_aut.sas7bdat	DVDECOD	5	017924-604-00024	Concomitant Medications/<_carriage return_><_line feed_>Therapy	<_carriage return_><_line feed_>

Output 3. Excel output generated by 'PD_list_ASCII.sas'

Based on the output, Ping knows: in automatic PD dataset located in 'DEVOPS/NCR17375/ACC17923/CDM_DEVIAT/SDTM/DATA ACC17923_bs_aut.sas7bdat', two special characters are found in DVDECOD variable: 'carriage return'+ 'line feed' in observation 5.

Now it is clear, special character in DVDECOD comes from PD list and copied into PD matrix by Ping, that's why it is fetched by both macros. Special character in DVCM comes from manual PD dataset which is entered by Singh and extracted by Ping, that's why it is not fetched by 'PD_list_ASCII.sas'.

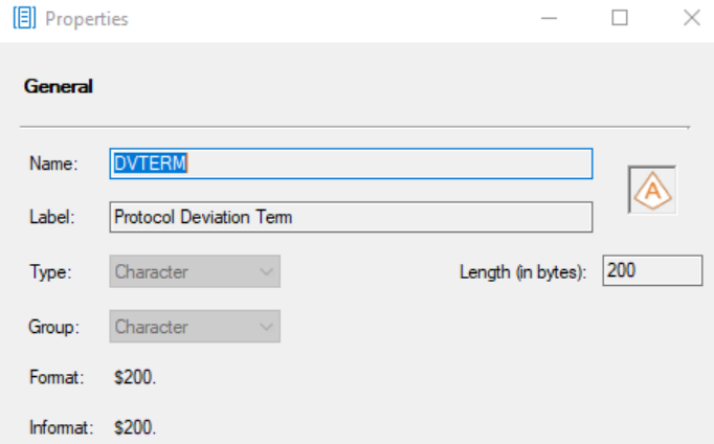
Checking program

Now let's look at which method is used by Ping to fetch the special character. Below codes comes from 'SAS_output_ASCII.sas':

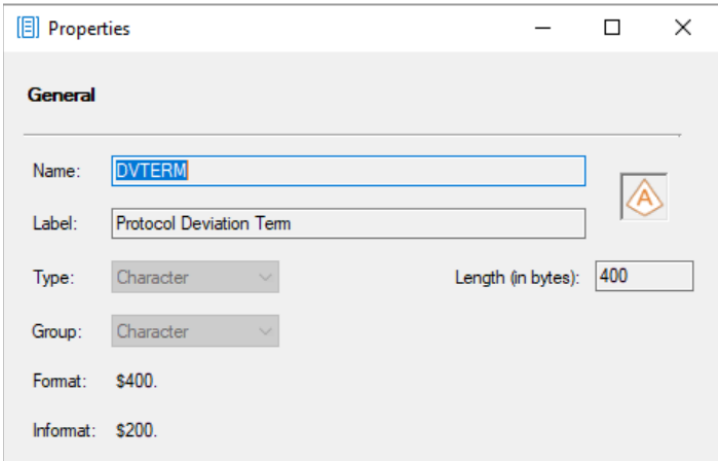
```
libname indata
"&W_ROOT./&W_MODWORK.OPS/&W_COMPOUND./&W_STUDY./&W_ANALYSIS./SDTM/DATA"

access=readonly inencoding=any cvpmultiplier=2;
```

Here Ping use SAS CVP (Character Variable Padding) engine to specifies a multiplier value to expand character variable lengths when you are processing a SAS data file that requires transcoding. The CVP engine expands the lengths to avoid character data truncation risk. The lengths for character variables are increased by multiplying the current length by the specified value. You can specify a multiplier value from 1 to 5, or you can specify 0 and then the CVP engine determines the multiplier automatically. If CVPMULTIPLIER=2, then character variable lengths are increased by multiplying the lengths by 2, i.e., a character variable with a length of 100 is increased to 200.

Display 1 and display 2 shows the length change by SAS CVP engine.



Display 1. DVTERM variable length in import file before read by SAS CVP engine



Display 2. DVTERM variable length in new dataset derived by SAS CVP engine

You can also use CVPBYTES which operates by expanding the character variables by the number of bytes specified (i.e., adding specified bytes to variable length, bytes values should be from 0 to 32766), just make sure the number of bytes you specified must be large enough to accommodate any expansion. If you explicitly specify the CVP engine but do not specify either CVPBYTES= or CVPMULTIPLIER=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths.

Next step, Ping use below codes to generate a list of non-special characters (ASCII printable characters):

```
data ListOfASCII;
  length ASCII_List $200.;
  ASCII_List='';
  do i=32 to 126;
```

```
    ASCII_List=trim(ASCII_List)||byte(i);
  end;
run;
```

The BYTE(n) function specifies an integer that represents a specific ASCII character. For example, if n is assigned as 33, then "!" will be returned by BYTE(33) because 33 is the decimal value of '!' in ASCII. Figure 5 shows the value of 'ASCII_List' varible:

Figure 5. The value of 'ASCII_List' varible display all ASCII printable characters (decimal value 32-126)

Now it's easy to fetch all special characters based on above list using below codes:

```
Special_Character=kcompress('||name||',ASCII_List);'
```

The KCOMPRESS function specifies a character expression that contains the characters to be removed, so the 'Special_Character' value will be assigned if any special character existed in the imported file. Then the excel output can be generated based on above finding.

SPECIAL CHARACTER CORRECTION

Now Ping gets the special character lists, next challenge is how and who to correct them, and how to avoid the similar issue in the future?

Alignment with downstream role

As we explained at the beginning, we need to confirm the definition of 'Special character' and align with all stakeholders. For example, if the clinical trial is sponsored by a pharma company in eastern European country, and the LATIN2 (extended ASCII Table for ISO-8859-2) is allowed and preferred to be used in submission datasets to support Polish (e.g., ą - Latin small letter a with ogonek) by company and local agency. In such case, BP need to follow guidance to generate SDTM and ADAM based LATIN2 encoding, however if DMP use above method to fetch, remove, or replace all special characters which is not belongs to LATIN1 without align with BP, some import data with Polish may be lost or misinterpreted in submission files.

To avoid above risk, Ping confirmed with BP team leader, and get a team level consensus: DMP and BP will follow the same rule that consider all characters not in ASCII printable characters (decimal value 32-126) as special character.

Manual update VS Automatic update

Now let's discuss how and who should correct the special characters. Table 4 shows the pros and cons of manual VS automatic (by program) way to update (remove or replace) the source file:

	Manual	Automatic
Pros	1. Flexible 2. High quality 3. Consistent between source file and deliverable	1. High efficiency 2. High consistency in deliverable quality
Cons	1. Low efficiency 2. Inconsistency in deliverable quality 3. May forgot to check	1. Cannot handle complex case like polysemous words (e.g., tête in French can means human head in anatomy, or top of a medical instruments) 2. Inconsistent risk if source file cannot be updated accordingly

Table 4. Manual update VS Automatic update of the source file

It's teams' decision to choose the checking method considering resource and department / company requirement, but no matter which method is selected, we must have a convention rule to indicate how to remove or replace special characters, figure 6 shows the special characters convention rule drafted by Ping:

No.	Special char	Special char ASCII Code (0 to 31, 127 (control characters), 128 to 255 (extended ASCII/upper ASCII: WLATIN1); 0-127 is lower ASCII: LATIN1)	Special char description	Suggested char	Suggested char ASCII Code (printable characters, 32-126)	Suggested char description
1	–	150	En dash	-	45	Hyphen-minus
2	—	151	Em dash	-	45	Hyphen-minus
3	²	178	Superscript two - squared	^2	94, 50	Caret - circumflex, 2
4	³	179	Superscript three - cubed	^3	94, 51	Caret - circumflex, 3
5	Ü, etc.	220	Latin capital/small letters	U, etc		Ask team to translate into english word
6	©	169	Copyright sign		NA	Remove it
7	®	174	Registered trade mark sign		NA	Remove it
8	°	176	Degree sign	degree		Replaced by english word
9	±	177	Plus-or-minus sign	+-	43, 45	Plus, hyphen-minus
10	•	149	Bullet	1), etc.	49, 41	1, Close parenthesis (or close bracket)
11	µ	181	Micro sign	u	117	Lowercase u
12	·	183	Middle dot - Georgian comma		32	Space
13	……	NA	Chinese ellipsis (shift+^ in Chinese mode)	...	46	Horizontal ellipsis (period (dot or full stop) enter three times)
14	mm^2, mm^3, etc.	NA	square/cubic millimeters	mm2, mm3, etc.		Superscript replaced by numeric
15	≤	NA	Less than and equal to	<=	60, 61	Less than, equal to
16	≥	NA	Greater than and equal to	>=	62, 61	Greater than, equal to
17	'	NA	Apostrophe	'	39	Single quote
18	α, etc.	NA	Greek letters	alpha, etc.		Replaced by english word
19	é, etc.	NA	French letters/other language letters which is not in LATIN 1			Ask team to translate into english word
20	HT	9	horizontal tab		NA	Remove it
21	VT	11	vertical tab		NA	Remove it
22	CR	13	carriage return		NA	Remove it
23	LF	10	NL line feed, new line		NA	Remove it
*Also avoid: leading space; only dot (.) in character field						

Figure 6. Special characters convention rule

This table list the most used special character and corresponding correction suggestion, including suggested counterpart that would be an acceptable replacement based on DMP's working experience, as well as the character description and corresponding ASCII code (decimal value) if applicable.

You can see some complex scenarios in row 5 and 19, team decision is required to decide which English word should be used for replacement. Considering that, Ping decides to choose manual way to make sure the high quality of her deliverables.

To minimize the cons of manual check, Ping also developed a PD check list which is mandatory to be completed for all DMPs with below purpose:

1. Remind every DMP to follow all required steps at corresponding stage during entire PD activities.

2. Keep consistency in quality, make sure every DMP is at the same page and take the same action.

Figure 7 shows the instruction tab of PD check list.

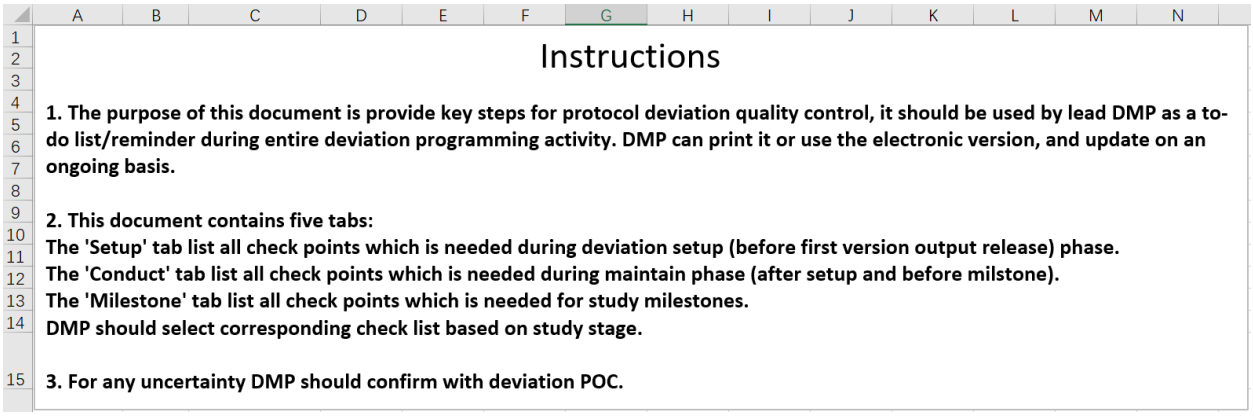


Figure 7. PD checklist instruction tab

This check list also answers our previous question that who will take responsibility of manual check as below:

If any special character is fetched in automatic PD output, Ping should check PD matrix firstly:

1. If the character is not come from PD list, Ping should update PD matrix and spec directly.

2. If the character come from predefined list, Ping should also update matrix and spec directly, and remind SMM Maiko about this issue and make update suggestion based on special characters convention rule to make it consistent with PD matrix and spec. Remark: Maiko is a medical expert with limited programming knowledge, so Ping need to explain rational patiently.

If any special character is fetched in manual PD, DMP alert Study Monitor (SM) Singh to update the related record in manual PD collection system.

In a summary, the overall principle is:

1. Who enter the special character who correct it, the file owner takes the final responsibility to make sure all special characters are excluded in their files.

2. DMP and BP take the responsibility to check special character in their deliverables by program and remind upstream role in case of any finding.

3. To avoid the special character as much as possible from the origin, the best practice is documenting the definition, harmfulness, and good practice of handling special character in related quality document and template (e.g., PD list, PD spec, and PD matrix instruction for automatic PD; manual PD collection system job aid for manual PD). It also would best to highlight the risk by holding a series of transversal meetings with related department representatives.

EMERGENCY HANDLING

Finally, let's discuss the approach in some emergency. As we discussed previously, manual update is the best way to guaranty to data with highest quality, but please imagine below scenario:

Partial database lock activity is planned for tomorrow, but DMP just take over the study one week ago, today he found hundreds of rows in manual PD have a same special character – (En dash; decimal value 150) in DVSPID variable which is required variable in SDTM and ADAM. After checking with study team imminently, he knows previous DMP did not follow the check list to fetch the special characters, and this issue is cause by a manual PD collection system bug, however, monitor team confirm they have limited resource to manually update all the DVSPID variables in such urgent deadline.

All stakeholders get an alignment via an ad-hoc meeting: study team will record this issue into a dedicated issue tracking log, and DMP will remove all the En dash special character by program to make sure SDTM and ADAM will not be impacted.

In such scenario, eliminating the source of the special character problem is not a possibility, DMP should evaluate the plan B to development a macro to replace the characters with appropriate ASCII equivalents automatically.

As previously mentioned, DMP can use BYTE function to explicitly identify the decimal value of characters. Besides, below methods can also be used depending on different purpose:

Method I:

The function PRXCHANGE: using Perl regular expressions to specify and replace special characters by its hexadecimal value.

Method II:

The function COMPRESS: returns a character string with specified characters removed from the original string.

Method III:

The function TRANSLATE: replaces specific characters in a character expression, designed for SBCS data.

Or the function KTRANSLATE: same as TRANSLATE but designed for use with MBCS.

As a summary:

1. Automatic way can be used in urgent situation with the major advantage of high efficiency and high result consistency, but the method should carefully investigate to avoid unexpected update.

2. This approach may bring some risks such as inconsistency between source file and result, so well document this action is necessary for compliance and audit / inspection consideration.

3. This approach can only handle some simple and scripted situation, for complex case which need human judgement still need manual check for quality consideration.

CONCLUSION

Identifying and resolving special character issues is a common task during DMP's daily work. To effectively handle this task, four key capabilities are required:

1. A thorough understanding of special characters and the causes of their occurrence: including the definition of special characters, the various ways they can arise, and the potential impact on data processing and submission.

2. Awareness of the difference between automatic and manual approaches in addressing special character issues: knowing the pros and cons of them enables DMP to make informed decisions and choose the most suitable approach for specific situations.

3. Familiarity with commonly used special character handling programming functions: DMP should have a comprehensive understanding of commonly used functions to process special characters, and the strength and limitations for each one and their applicability in different scenarios.

4. Good communication skills with both upstream and downstream users: it enables DMP to get information about the root cause of special character issues, better facilitate the implementation of appropriate solutions, and establish an effective preventive strategy to limit the possibility of special characters entered at the beginning.

By cultivating these four capabilities, DMP can effectively identify, analyze, and resolve special character issues as early as possible to prevent any last-minute disruptions before clinical data submission.

REFERENCES

List of Unicode characters. https://en.wikipedia.org/wiki/List_of_Unicode_characters#Latin_script.

The Unicode Consortium. The Unicode Standard, Version 15.1.0, (South San Francisco, CA: The Unicode Consortium, 2023. ISBN 978-1-936213-33-7). https://www.unicode.org/versions/Unicode15.1.0/

ASCII history. https://en.wikipedia.org/wiki/ASCII

ASCII Characters. https://www.ascii-code.com

SAS online documentation. https://support.sas.com/en/documentation.html

Draft Guidance for Industry Request for Quality Metrics, published July 28, 2015. https://www.fda.gov/media/98939/download

ACKNOWLEDGMENTS

I would like to express my gratitude to the people who helped me to write this paper: Weina Jia for the encouragement and inspiration. Shelly Jin, April Zang, Tia Yang, Yuhan Li for the review and comment upon the strategy. Qiying Yang for testing the SAS code.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Fei Guo
Sanofi
5 Floor Sanofi, HP building, No. 112 Jian Guo Road, Chaoyang District, Beijing, P.R. China.
fei3.guo@sanofi.com