



RAIN Free-Form Encoding Schemes

(Issued: 2025-12-15)

NOTE: The RAIN Free-Form encoding schemes do not ensure global item identifier uniqueness (global uniqueness of the UII prevents inter-application interference in all read-scenarios). RAIN's extended read range and capability to read multiple tags "simultaneously" are powerful features, but they can result in Tag Clutter (inter-application interference) if tags are not encoded uniquely. To ensure uniqueness for non-GS1 compliant applications, the user is recommended to obtain a Customer Identification Number (CIN) from an approved ISO/IEC 15459 Issuing Agency, like the RAIN Alliance, and use it with the appropriate AFI. Using AFIs 0xAE or 0xBC together with a RAIN Alliance-issued CIN provides an efficient means for non-compliant encodings to achieve global uniqueness and standards compliance (see [RAIN Alliance-Issued CIN with Free-Form Encoding Schemes \(BINARY and TEXT\)](#)).

1 Purpose of the RAIN Free-Form encoding schemes

The RAIN Alliance has become aware of a proliferation of applications which encode hex, text and other strings (essentially, "Free-Form encodings") in RAIN tags without setting the item identifier type bits (Toggle-bit and AFI) in the PC-Word. This is done because of the following reasons:

1. Lack of awareness and understanding of item identifier standards.
2. Lack of awareness and understanding of RAIN Tag Clutter (inter-application interference).
3. A move from barcode/NFC text encodings to take advantage of RAIN capabilities, like range and multi-tag reading.
4. An erroneous belief that only "my" tags will be present in my application's read-zones.
5. An erroneous belief that my tags will not enter the read-zones of other applications.
6. Using the RAIN tag as a key fob, i.e. as a keyboard input, often in manufacturing, to avoid operator errors and to improve operator efficiencies. This is a common practice using barcodes and NFC. RAIN tags have now also been used, not only as a key fob, but also to manage items.

EXAMPLE: A casting manufacturer transitioned from using text in barcodes to identify moulds, to using RAIN. RAIN enabled the factory to manage and find the dies when in storage. This allows for higher efficiencies in short production runs since the setup time is reduced by finding the dies more quickly. However, the casting machine uses a keyboard input for the dies used in a cast to check consistency and interoperability of the dies, as done with the barcodes. As such text is encoded in the tag.



7. Companies are reluctant to change deployed data systems.

The result is that these non-compliant Free-Form encodings conflict with compliant encodings and lead to Tag Clutter, especially in environments rife with GS1 EPC encoded tags where the T bit is set to 0₂, the default encoding for many RAIN tags.

Utilizing the AFIs designated for the RAIN Free-Form encoding schemes means that non-compliant, closed-loop systems only need to set the T-bit to 1₂ and encode the appropriate AFI.

Data presentation	AFI for MB01 (Ull)	DSFID for MB11 (UserMem) ¹
Free-Form BINARY	0x01	33 (0x21)
6-bit TEXT	0x02	34 (0x22)
UTF-8 TEXT	0x03	35 (0x23)

¹ The DSFID is used to add Free-Form data encoded to MB11 (UserMem), to a Ull.

Additional data to the free-from encodings may be encoded in MB11 using the appropriate DSFID.

EXAMPLE: A URI may be encoded in MB11, which can be prefixed to the Ull. See the RAIN Alliance specification *RAIN URI Identifier*.

Usage of these AFIs dramatically lowers the hurdle to become compliant for existing non-compliant Free-Form binary and text encodings, as well as provides a path to RAIN without upsetting deployed processes and systems.

NB: All encoding schemes outlined in this document are limited in use to closed-loop applications and environments.

2 Terms

Term	Description	Specification
0xH...H	A hex presented binary string, with 'H' representing a '0' to '9' and 'A' to 'Z'.	ISO/IEC 9899
AFI	Application Family Identifier, which is part of the PC-word.	ISO/IEC 19762
DSFID	Data Storage Format Identifier.	ISO/IEC 15961
Free-Form	An issuer-defined data encoding where the structure and format are not fixed by a standard, but are freely chosen by the issuer	Oxford English Dictionary — “Free-Form: not following a regular or conventional structure.”
Gen2V2 Gen2V3	Versions of the GS1 EPC® <i>Radio-Frequency Identity Generation-2 UHF RFID Standard</i> .	Gen2V3 is fully incorporated in ISO/IEC 18000-63.
MBxx	Memory Banks 00, 01, 10 and 11 of a RAIN tag.	ISO/IEC 18000-63
PC-Word	Protocol Control word (16 bits) which is transmitted leading the Ull. Provides metadata about a tag’s encoding.	ISO/IEC 18000-63



Ull	Unique Item Identifier. The ISO equivalent to GS1's term "EPC".	ISO/IEC 19762
UserMem User Memory	MB11 of a RAIN tag.	ISO/IEC 18000-63
UTF-8	Unicode Transformation Format, 8-bit	ISO/IEC 10646, RFC 3629
X...X	A bit presented binary string with 'X' representing a '0' or '1'.	-

3 AFI 0x01: 16-bit Free-Form BINARY

3.1 Ull format and encoding

When the AFI is set to 0x01, then the Ull shall be encoded with a multiple of 16-bit words. Any input character set and supported character-to-bit conversion may be used. BINARY encodings / readings are generally represented using hex to transmit the payload.

3.2 MB11 (UserMem)

DSFID 33 (0x21) shall be used to encode Free-Form BINARY to MB11 for all AFIs, subject to the specification of the AFI.

4 AFI 0x02: 6-bit Free-Form TEXT

4.1 Ull format and encoding

When the AFI is set to 0x02, then the Ull shall be encoded using the ISO/IEC 15434 direct encoding character set as specified in ISO/IEC 15962, see Annex A.

The TEXT string bit encoding shall be padded with zero bits to a 16-bit boundary.

NOTE The group separator character "⁶s" can be used to separate a multipart TEXT string.

4.2 MB11 (UserMem)

DSFID 34 (0x22) shall be used to encode Free-Form 6-bit text characters to MB11 for all AFIs, subject to the specification of the AFI.

The TEXT string bit encoding shall be padded with zero bits to a 16-bit boundary.

NOTE The group separator character "⁶s" can be used to separate a multipart TEXT string.

5 AFI 0x03: UTF-8 (8-bit multiples) Free-Form TEXT

5.1 Ull format and encoding

When the AFI is set to 0x03, then the Ull shall be encoded as UTF-8 (ISO/IEC 10464 and RFC 3629).

The Ull shall be terminated with the 0x00 byte, or the last byte of the last word of the Ull as indicated by the L-bits in the PC-Word.

NOTE The group separator character "␣" can be used to separate a multipart TEXT string.

5.2 MB11 (UserMem)

DSFID 35 (0x23) shall be used to encode Free-Form UTF-8 text characters to MB11 for all AFIs, subject to the specification of the AFI.

5.3 Ull decoding and data presentation

When reading a tag of which the T-bit is set to 1₂ and the AFI is set to 0x03, then the reader shall report:

1. T=1₂, AFI = 0x03
2. Ull = The UTF-8 string up to the 0x00 byte or the end of the Ull read.

EXAMPLE "RAIN is cool 🍌" which is encoded as follows:

```
T = 12
AFI = 0x03
Ull = D2 41 49 4E 20 69 73 20 63 6F 6F 6C F09F918Fhex
      R A I N _ i s _ c o o l 🍌
```

'_' indicates the space character.

The total amount of bits is 120 with 🍌 and 88 without.

5.4 Extending the Free-Form Ull with MB11

The rules to extend the UTF-8 string in User Memory (MB11) are:

1. The AFI shall be set to 0x03 and the DSFID shall be set to 0x23 encoded as the first byte (most significant byte) of the first word of MB11, word 0.
2. The UTF-8 encoding shall start at the second byte of the first word of MB11, which is word 0.
3. The UTF-8 encoding shall be terminated with the zero byte 0x00, or the end of the tag's MB11.
4. "␣" may be used to separate a multipart UTF-8 string.



5. For Gen2V3 tags the UWC (User Memory Word Count) may be set to allow for the tag to declare it has a UTF-8 string in User Memory.

Reading the UTF-8 string from User Memory involves:

- Gen2v2 readers: When the UMI-bit ("there is User Memory") is set in the PC-Word and received during Inventory, and the reader is configured to read User Memory, then the reader shall read User Memory to the first word containing the 0x00 byte or end-of-memory, and present the UTF-8 string up to the 0x00 byte or end of memory.
- Gen2v3 and tags: When the RUM-bit (Read User Memory) is set, then the reader should read the User Memory with the ReadVar(UM,0,0) command. The tag then responds with the User Memory words as set by the UWC (User Memory Word Count).

The reader presents the User Memory UTF-8 string as is.

EXAMPLE 1 UII = "RAIN is cool 🍹" and UserMem = "for all purposes!"

The reader may append the UTF-8 string read from User Memory to the UII, using the separator "␣".

EXAMPLE 2 "RAIN is cool 🍹␣for all purposes!"

Annex A (normative)

The ISO/IEC 15962 6-bit character for AIDC use

6-bit #	Character	6-bit #	Character	6-bit #	Character	6-bit #	Character
100000	[space]	110000	0	000000	@	010000	P → p
100001	<EOT> ^{e_t}	110001	1	000001	A → a	010001	Q → q
100010	"	110010	2	000010	B → b	010010	R → r
100011	<FS> ^{f_s}	110011	3	000011	C → c	010011	S → s
100100	<US> ^{u_s}	110100	4	000100	D → d	010100	T → t
100101	%	110101	5	000101	E → e	010101	U → u
100110	&	110110	6	000110	F → f	010110	V → v
100111	'	110111	7	000111	G → g	010111	W → w
101000	(111000	8	001000	H → h	011000	X → x
101001)	111001	9	001001	I → i	011001	Y → y
101010	*	111010	:	001010	J → j	011010	Z → z
101011	+	111011	;	001011	K → k	011011	[
101100	,	111100	<	001100	L → l	011100	\
101101	-	111101	=	001101	M → m	011101]
101110	.	111110	>	001110	N → n	011110	<GS> ^{g_s}
101111	/	111111	?	001111	O → o	011111	<RS> ^{r_s}