PRACTICAL EXAMPLES & FAQ

1. Practical Examples using the RAIN Alliance ISO Numbering System

Use of access or kill passwords, user memory, locking, permalocking, and other advanced RAIN RFID features are not covered in this document. In most cases, these additional tag features do not impact selection of a numbering system.

Example 1:
I have an existing barcode-based asset tracking system that uses a prefix of 4 letters, follow by 6 numbers. These assets never leave my organization, there are 10,000 of them currently and I add ~100 more each month. Example asset ID numbers are: ABYZ123456, CDWX789012

For a deployment of this size, choose a 6-digit / 24-bit RAIN CIN length. We'll assume an RFID tag with 128 bits EPC/UII memory, no User Memory, and no XPC features.

First, select the PC Word value for: Length= 8 words (128 bits), UMI=0_2, XPC=0_2, T=1_2 AFI=0xAE. From the table, that would be 0x41AE.

We'll assume the assigned CIN from RAIN is 123456, which in hexadecimal is 0x87C440.

With 128 bits EPC/UII memory available, and a RAIN CIN of 24 bits, that leaves 104 bits for my asset data. But, first we have to decide how to store the alphanumeric data in hexadecimal or binary.

One option is to use 8-bit ASCII encoding (https://en.wikipedia.org/wiki/ASCII#8-bit_codes). This approach assigns each number, uppercase and lowercase letter, and some punctuation characters an 8 bit or 2 Hexadecimal character code. Many RAIN RFID readers, printer/encoders, and software tools can convert ASCII characters to hexadecimal for either encoding or reading tag data. There are also many on-line tools available for quick data conversion.

With 10 alphanumeric characters in my asset IDs, converting them to hexadecimal will result in 80 bits or 20 hex characters. But, I have 104 bits of available data, which is enough space for 13 characters. In this example, we'll just pad the original asset IDs with some extra zeros. Another option would be to use a shorter length value in the PC word.

ASCII Asset ID	Hexadecimal Asset ID (80 bits)
ABYZ123456	0x4142595A313233343536
CDWX789012	0x43445758373839303132

ASCII Asset ID with padded zeros	Hexadecimal Asset ID (104 bits)
ABYZ123456000	0x4142595A313233343536303030
CDWX789012000	0x 43445758373839303132303030

Now I can construct the complete EPC/UII:

PC Word	RAIN CIN Header	Asset ID Hexadecimal	Original Asset ID
0x41AE	0x01E240	0x4142595A313233343536303030	ABYZ123456
0x41AE	0x01E240	0x43445758373839303132303030	CDWX789012

Example 2.
I run a record keeping system for a large law firm. We need to track more than 1 million documents per year. Our current document IDs are 16 characters long and alphanumeric. Example Document ID numbers are: 1234ABCD-4567EFGH and WXZY1234-QRST5678

For a deployment of this size, choose a 4-digit / 16-bit RAIN CIN length. Example 1 used an 8-bit ASCII encoding scheme, but with 17 characters (including the dash) that would require 152 bits -- 136 bits for the data and 16 bits for the CIN. While there are tag ICs available with that amount of EPC/UII memory, this example will show how to use Base36 encoding (https://en.wikipedia.org/wiki/Base36) and a more commonly available 128 bit size EPC/UII.

Like example 1, assume an RFID tag with 128 bits EPC/UII memory, no User Memory, and no XPC features will be used. Also assume the assigned 4-digit CIN from RAIN is 1234, which in hexadecimal is 0x8952. Before deciding on the PC Word, we need to determine how to encode the document IDs.

Base36 uses the numbers 0-9 and capital Latin letters A-F. It requires <6 bits per character, and for RFID encoding in hexadecimal it works best for data lengths that are a multiple of 8. In this example we'll encode our 16-digit document IDs in 2 groups of 8. Conversion between hexadecimal and Base36 is not as widely supported as ASCII by RFID software, but it's a good option encoding scheme for alphanumeric data in RAIN RFID tags.

It's straight forward to convert between Base36 using an on-line tool, excel, or programming script. Each group of 8 Base36 characters are converted to 12 hexadecimal characters, which takes 96 bits total. With our 16-bit RAIN CIN, that makes the total data length 112 bits. Select the PC Word value for: Length= 7 words (112 bits), UMI=0_2, XPC=0_2, T=1_2 AFI=0xAE. From the table, that would be 0x39AE.

PC Word	RAIN CIN Header	Asset ID in Hexadecimal	Document ID in Base36
0x39AE	0x04D2	0x134D9A27ED 4B9A91D4C1	1234ABCD 4567EFGH
0x39AE	0x04D2	0x25916F59ED0 1E8798E0BA4	WXZY1234 QRST5678

When reading the tags, make sure to use the same process in reverse – converting the hexadecimal Asset IDs back into Base36 in the same groups of 12 characters.

Example 3.
I want to track company IT assets and both a 6-digit alpha numeric asset ID and a 48-bit device MAC address on a 128 bit RAIN RFID tag. I expect to use about 500 tags per year. Examples asset IDs are ABC123 and 789XYZ. Example MAC addresses are 00-14-22-01-2C-45 and 00-40-96-A4-F1-34.

For a deployment this size, choose an 8 digit / 32-bit CIN length. The 6-digit asset IDs can be encoded using 8-bit ASCII with 48 bits, followed by the 486bit MAC address which is already represented in hexadecimal.

Assume we're using a tag with 128 bits of EPC memory, 32 bits of User memory, and no XPC features. Select the PC Word value for: Length= 8 words (128 bits), UMI=1_2, XPC=0_2, T=1_2 AFI=0xAE. From the table, that would be 0x45AE.

We'll assume the assigned CIN from RAIN is 12345678, which in hexadecimal is 0x85F1C24E.

PC Word	RAIN CIN Header	Asset ID + MAC in Hexadecimal	Asset ID	MAC Address
0x45AE	0xBC614E	0x414243313233 001422012C45	ABC123	00-14-22-01-2C-45
0x45AE	0xBC614E	0x37383958595A 004096A4F134	789XYZ	00-40-96-A4-F1-34

Example 4.
I run a logistics business with parcel volume exceeding 100 million packages per year. How can I use the RAIN ISO numbering system to identify packages within my network?

For a deployment this size, choose a 2 digit / 8-bit CIN length. With a commonly available tag with 128 bits of EPC/UII memory, that would leave 120 bits to encode a package IDs. If used efficiently 120 bits can store 2^{120} or 1.3×10^{36} unique numbers – which is an unfathomably large number!

With the RAIN ISO number system, these 120 bits can be encoded using any method desired. One commonly used approach is to divide the space into sections that represent different attributes. For example, you might choose to segment the EPC/UII data like this:

	RAIN CIN	Service Level	Time Stamp	Origin Code	Destination Code	Package Identifier
Encoded Length	8 bits	8 bits	24 bits	20 bits	20 bits	48 bits
Numbers available	90	256	~16.7 million	~1 million	~1 million	~280 trillion
Purpose	Identifies tag owner	Freight, Ground, Air, Overnight, Priority, etc.	Identify every minute since 1/1/2022 for ~32 years	Identify every US ZIP code, with room to grow		Uniquely identify 280 trillion packages

Again assume we're using a tag with 128 bits of EPC memory, 32 bits of User memory, and no XPC features. From the table select the PC Word value for Length= 8 words (128 bits), UMI=1_2, XPC=0_2, T=1_2 and AFI=0xAE, which is 0x45AE. We'll assume the assigned CIN from RAIN is 12, which in EBV-8 hexadecimal is 0x0C.

For a package shipped at 8am on October 18, 2022 from Chicago ZIP code 60610 to Denver ZIP code 80202 with service level of 0xA5, example tag data might look like this:

PC Word	RAIN CIN Header	Service Level	Time Stamp	Origin Code	Destination Code	Package Identifier
0x45AE	0x0C	0xA5	0x066120	0x0ECC2	0x1394A	0x123ABC456DEF

2. Frequently Asked Questions

I encoded ASCII data to my tag, so why is my reader showing me something different?
Most RAIN RFID readers and reader software report tag data in Hexadecimal by default. If you've encoded data using the RAIN ISO numbering system in ASCII, you'll need to convert the data after the CIN header from hexadecimal back into ASCII.

Why isn't my reader showing me the PC word?
Some RAIN RFID readers and reader software don't show the PC Word by default. Consult with your reader or software provider to make sure they can support filtering tags by the PC Word.

Why can't I encode a shipping address in an RFID tag?
RAIN RFID tags have limited amount of memory, the most widely used only have ~128 bits of encodable memory. While that's enough space to uniquely identify a mind-bogglingly large number of items, it's only enough for 16 ASCII characters.

How do I convert between Decimal, Hexadecimal and Binary?
There are many tools available, including Microsoft Excel conversion functions, the Windows calculator (in programmer mode), or numerous on-line tool

3. Table of common PC Word Values using the RAIN AFI

The following table shows valid PC word values in binary and hexadecimal for Toggle Bit = 1_2, XPC = 0_2, AFI = 0xAE, and UMI = 0_2 or 1_2.

	UMI = 0		UMI =1	
EPC/UII Length (bits)	**PC Word (Binary)**	**PC Word (Hexadecimal)**	**PC Word (Binary)**	**PC Word (Hexadecimal)**
0	00000001	01AE	00000101	05AE
16	00001001	09AE	00001101	0DAE
32	00010001	11AE	00010101	15AE
48	00011001	19AE	00011101	1DAE
64	00100001	21AE	00100101	25AE
80	00101001	29AE	00101101	2DAE
96	00110001	31AE	00110101	35AE
112	00111001	39AE	00111101	3DAE
128	01000001	41AE	01000101	45AE
144	01001001	49AE	01001101	4DAE
160	01010001	51AE	01010101	55AE
176	01011001	59AE	01011101	5DAE
192	01100001	61AE	01100101	65AE
208	01101001	69AE	01101101	6DAE
224	01110001	71AE	01110101	75AE
240	01111001	79AE	01111101	7DAE
256	10000001	81AE	10000101	85AE
272	10001001	89AE	10001101	8DAE
288	10010001	91AE	10010101	95AE
304	10011001	99AE	10011101	9DAE
320	10100001	A1AE	10100101	A5AE
336	10101001	A9AE	10101101	ADAE
352	10110001	B1AE	10110101	B5AE
368	10111001	B9AE	10111101	BDAE
384	11000001	C1AE	11000101	C5AE
400	11001001	C9AE	11001101	CDAE
416	11010001	D1AE	11010101	D5AE
432	11011001	D9AE	11011101	DDAE
448	11100001	E1AE	11100101	E5AE
464	11101001	E9AE	11101101	EDAE
480	11110001	F1AE	11110101	F5AE
496	11111001	F9AE	11111101	FDAE