BOWH RFID SERVER POS INTEGRATION SPECIFICATION

Issue No.	0.5
Issue Date	03/15/08
Number of Pages	8
Author	Stanley Li
Document Filename	Bowh_RFID_Server_POS_Integration_Spec_0.5.doc

Contents

1. Document Control

1.1. Sign Off

Name	Company	Signature / Date Sign-off received
Matt Sipos	Bowh Inc	
Engineer	POS company	

Subsequent changes to this document are through the Exception Process

1.2. Change History

Issue Number	Issue Date	Revised by	Reasons for Update
0.1	9th Nov 07	Stanley Li	First Draft.
0.2	9th Nov 07	Stanley Li	Editing Revision.
0.3	14th Nov 07	Stanley Li	Add UPC <-> EPC convention. Update the inventory table with quantity.
0.4	14th Nov 07	Stanley Li	Add UPC <-> EPC conversion examples.
0.5	5th Mar 08	Stanley Li	Add alphabetic support for EPC UPC conversion.

1.3. REFERENCE DOCUMENTATION

Reference No.	Document Name

2. Purpose

This document is intended to define a uniform way for Bowh RFID Server software to access data from a POS database. It also defines the conversion between UPC code and EPC code.

3. Data Sync

Bowh RFID Server will directly poll product and inventory data from POS database tables. The POS database needs to implement two (2) new tables (Product_B and Inventory_B) for Bowh.

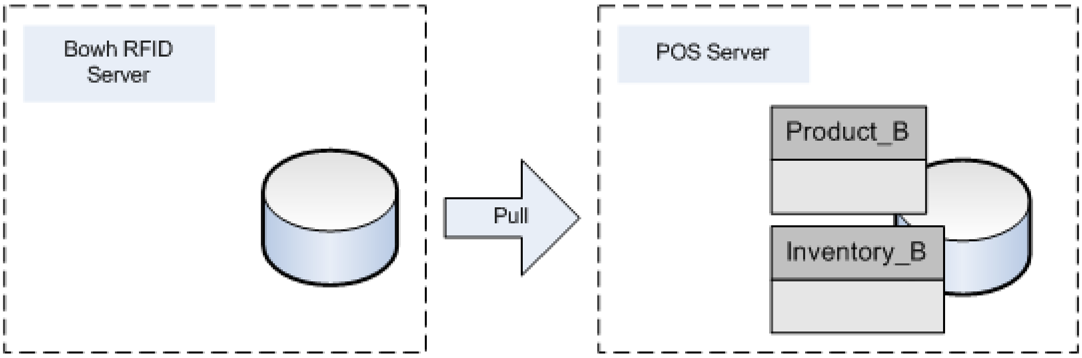

Figure 1: Database Sync

The benefit of the above two table approach is to minimize the risk of unwanted operations damaging the integrity of the POS database whilst limiting traffic between the two databases.

4. Table Definition

4.1. Product Table (Product_B)

The product table is used to store all the products that have been defined in the POS system.

Field Name	Description	Type
ProductID	The unique identifier of the product in POS database	int
ProductName	Name of the product	varchar(50)
ProductUPC	The UPC code that represents a product	varchar(50)
ProductPrice	Price of the product per unit.	varchar(50)
ProductCost	The actual cost of the product per unit.	varchar(50)
ProductImage	The image that represent the product	image
ProductDescription	The description of the product	varchar(MAX)

Figure 2: Product Table

4.2. Inventory Table (Inventory_B)

The inventory table contains all the items that are stored in inventory.

Field Name	Description	Type

Field Name	Description	Type
InvID	The unique identifier of the inventory item.	int
InvUPC	UPC code of the item	varchar(50)
InvSerial	Serial number of the item.	int
InvEPC	EPC code of the item.	varchar(50)
InvQty	The quantity of the item. Since the item is serialized, the qty can only be 1 or 0. “1” means the item is in-stock. “0” means the item is used to be in-stock but currently was sold.	int

Figure 3: Inventory Table

5. UPC <-> EPC Conversion Scheme

5.1. General Concepts

It is very important to make sure both Bowh RFID software and POS software have the same convention on converting UPC and EPC codes. The same convention being used will ensure once Bowh RFID software gets an EPC code from POS it knows the UPC code and vice versa.

The conversion scheme documented here is based on the following knowledge of RFID Gen2 tag and UPC standard:

<u>For general UPC (without any alphabetic characters):</u>

1) The EPC code is 12 bytes (96 bits) long.
2) The UPC code is a pure digit code and 14 digits in maximum (i.e. 12345678901234).
3) The EPC code should contain the UPC field (to hold the entire UPC code) and a concatenated serialized field (i.e. containing a serial number).

<u>For UPC with alphabetic characters:</u>

4) The EPC code is 12 bytes (96 bits) long.
5) The UPC code is a mix of alphabetic and numeric characters and 12 characters in maximum (i.e. ABCD12345678 or ABCDEFGHIJKL).
6) The EPC code should contain the UPC field (to hold the entire UPC code) and a concatenated serialized field (i.e. containing a serial number).

5.2. Conversion Scheme for pure numeric UPC

The suggested approach for the EPC code is as follows:

- The first byte is 0x30 as an identifier.
- The second byte is 0x00 for a reserved space.

- The third to eighth byte are used to store UPC (6 bytes).
- The last 4 bytes are used to store the serial.

For the UPC field, since it contains 6 bytes, it should be sufficient to hold the 14 digit (no alphabetical chars) UPC code. The last 4 bytes is able to hold a serial number of the item in a range of 0~4294967295.

	Header Field	Reserved Field	UPC Field	Serial Field
Capacity	1 Byte	1 Byte	6 Bytes	4 Bytes
Decimal Range	48	0	0~281474976710655	0~4294967295
Hex Range	0x30	0x00	0~0xFFFFFFFFFFFF	0~0xFFFFFFFF

Figure 4: EPC Scheme

5.3. CONVERSION SCHEME FOR ALPHABETIC MIXED UPC

The suggested approach for the EPC code is as follows:

- The first byte is 0x35 as an identifier.
- The second to tenth bytes are used to store UPC (6 bytes).
- The last 2 bytes are used to store the serial.

For the UPC field, since it contains 9 bytes, it should be sufficient to hold the 12 chars (including alphabetical) UPC code. Among these 9 bytes (72bits), each 6 bits should contain one alpha-numeric character. See following table for mapping of these 6 bits:

6 bits	Char	6 bits	Char	6 bits	Char	6 bits	Char	6 bits	Char
0	0	14	E	28	S	42	Reserved	56	Reserved
1	1	15	F	29	T	43	Reserved	57	Reserved
2	2	16	G	30	U	44	Reserved	58	Reserved
3	3	17	H	31	V	45	Reserved	59	Reserved
4	4	18	I	32	W	46	Reserved	60	Reserved
5	5	19	J	33	X	47	Reserved	61	Reserved
6	6	20	K	34	Y	48	Reserved	62	Reserved
7	7	21	L	35	Z	49	Reserved	63	Reserved
8	8	22	M	36	Reserved	50	Reserved		
9	9	23	N	37	Reserved	51	Reserved		
10	A	24	O	38	Reserved	52	Reserved		
11	B	25	P	39	Reserved	53	Reserved		
12	C	26	Q	40	Reserved	54	Reserved		
13	D	27	R	41	Reserved	55	Reserved		

Figure 5: 6 bit to alpha-numeric mapping table

The last 2 bytes is able to hold a serial number of the item in a range of 0~65536.

	Header Field	UPC Field	Serial Field
Capacity	1 Byte	9 Bytes	2 Bytes
Decimal Range	49	0~ 18446744073709551615	0~65536
Hex Range	0x35	0~0xFFFFFFFFFFFFFFFFF	0~0xFFFF

Figure 6: EPC Scheme with alphabetic UPC support

5.4. EXAMPLES

The following examples show how the conversion works between UPC and EPC codes.

1) UPC = 12345678901234, Serial = 1234

In this case the UPC is 14 digits. It is within the range of the capacity of the EPC and it is valid. The serial is also in the range of the capacity of EPC serial field. Please check the following figure on how this is converted -

	Header Field	Reserved Field	UPC Field	Serial Field
Decimal View	48	0	12345678901234	1234
Convert to Hex	0x30	0x00	0x0B3A73CE2FF2	0x000004D2

Figure 7: Example 1

Finally, if we concatenate all these 4 fields together, we have the converted EPC code – "0x30000B3A73CE2FF2000004D2"

2) UPC = 1234567890, Serial = 12345678

In this case the UPC is 10 digits, serial is 8 digits. Please check the following figure on how this is converted –

	Header Field	Reserved Field	UPC Field	Serial Field
Decimal View	48	0	1234567890	12345678
Convert to Hex	0x30	0x00	0x0000499602D2	0x00BC614E

Figure 8: Example 2

Finally, if we concatenate all these 4 fields together, we have the converted EPC code – “0x30000000499602D200BC614E”

3) UPC = 1234567890123456, Serial = 1234 (UPC Out of Range)

In this case, the UPC code contains 16 digits. This is out of range of current convention for a UPC capacity in an EPC code. So this won’t be converted.

4) UPC = 12345678901234, Serial = 12345678901 (Serial Out of Range)

In this case, the serial number contains 11 digits. This is out of range of current convention for a serial capacity in an EPC code. So this won’t be converted.