EPC-enabled RFID Serialization Management for SGTIN-96

GS1 US Guideline

GS1 US

THE GLOBAL LANGUAGE OF BUSINESS

Document Summary

Document Item	Current Value
Document Title	EPC-enabled RFID Serialization Management for SGTIN-96 GS1 US Guideline
Date Last Modified	9 May 2012
Current Document Issue	Issue #1
Status	Final
Document Description	

Contributors

Name	Organization
Gena Morgan (Group Facilitator)	GS1 US
Ken Traub (Editor)	Ken Traub Consulting LLC, for GS1 US
Joseph Andraski	VICS Staff
Paul Arguin	VICS Affiliates
Larry Arnstein	Impinj, Inc.
Susan Bailey	Hanes Brands
Kris Barton	Avery Dennison Corporation
Ray Blanchard	Avery Dennison Corporation
Daniel Bowman	Impinj, Inc.
Patricia Buccheri	GS1 US
Myron Burke	Wal-Mart Stores Inc
Phil Calderbank	SML
Michele Cervone	Maidenform
Kenmann Chiu	r-pac International Corp.
Thomas Considine	SML
Joe Demers	Zebra Technologies
Brian Derda	Macy's
Andy Edwards	Printronix Inc.
Patrick Ervin	Alien Technology
Michael Fein	Zebra Technologies
Harley Feldman	Seeonic, Inc.
David Feldman	Zebra Technologies
Jo Ann Fiordland	VICS Staff

Name	Organization
Susan Flake	Motorola
David Gardiner	VF Corporation
Doug Harvel	Jockey International, Inc.
Sue Hutchinson	GS1 US
David Isley	VF Corporation
Patrick Javick	GS1 US
Raj Jayaraman	Checkpoint Systems, Inc.
Steven Karrmann	JCPenney Company Inc
Peter Kim	Target Corporation
Mike Kuhno	Avery Dennison Corporation
Sri Ramya Kukunuri	Perry Ellis
John Mesa	Perry Ellis
Valerie Mitchko	GS1 US
Jim Musco	Macy's
Chris Nagy	Phillips-Van Heusen Corporation
Suresh Palliparambil	NXP Semiconductors
Fernando Perez	Perry Ellis
Dale Phillips	Hanes Brands
Sarah Polworth	Saks
Bebe Purcell	VF Corporation
Sheldon Reich	Cybra Corporation
Erin Saxton	Times Three Clothier, LLC
Paul Schmidt	Kohl's Department Stores Inc.
Michael Shabet	Cybra Corporation
Andrea Smith	Lord & Taylor
Kelly Snead	UPM RFID
Michele Southall	GS1 US
Loreta Stamo	Perry Ellis
James Stigall	UPM Raflatac
Pam Sweeney	Macy's
Michael Teitelbaum	r-pac International Corp.
William Toney	Avery Dennison Corporation
Ramon Valle	Perry Ellis
Victor Vega	NXP Semiconductors
Ismael Vicens	Wacoal America, Inc
Steve Voit	Impinj, Inc.
Gale Weisenfeld	Phillips-Van Heusen Corporation

Name	Organization
Kris Whitney	Macy's
McLeod Williamson	Zebra Technologies

Log of Changes in Issue #1

Issue No.	Date of Change	Changed By	Summary of Change
1	9 May 2012	GS1 US	Document created

Disclaimer

Whilst every effort has been made to ensure that the contents of this document are correct, GS1US and any other party involved in the creation of the document HEREBY STATE that the document is provided without warranty, either expressed or implied, of accuracy or fitness for purpose, AND HEREBY DISCLAIM any liability, direct or indirect, for damages or loss relating to the use of the document. The document may be modified, subject to developments in technology, changes to the standards, or new legal requirements. Several products and company names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

Table of Contents

1. Background 6
2. Scope 7
3. Business Scenarios 8
   - 3.1. Brand Owner Manufacturing on a Single Line 8
   - 3.2. Brand Owner Manufacturing on Multiple Lines 8
   - 3.3. Brand Owner Uses Service Bureaus, Contract Manufacturers, or Other 3rd parties 9
   - 3.4. Downstream (Distributor or Retailer) Exception Tagging 9
4. Approaches to Managing Serialization 10
   - 4.1. IT-Based Serialization 10
     - 4.1.1. Sequential Serialization on a Single Line 10
     - 4.1.2. Static Allocation of Serial Number Ranges to Multiple Lines 11
     - 4.1.3. Dynamic Allocation of Serial Number Ranges to Multiple Lines 14
   - 4.2. Chip Based Serialization (TID) 15
     - 4.2.1. About the TID 15
     - 4.2.2. Using the TID to Serialize Products 16
     - 4.2.3. Chip-based Serialization on Multiple Manufacturing Lines 18
     - 4.2.4. Considerations for Using Chip-based Serialization 18
   - 4.3. Creating a Top-level Serialization Plan 19
   - 4.4. Supporting Downstream Exception Serialization 21

1. Background

This guideline describes options and methods for assigning globally unique identification to individual instances of trade items, using a Global Trade Item Number (GTIN) plus a unique serial number. This combination is commonly referred to as a Serialized Global Trade Item Number, or SGTIN. Assigning a unique SGTIN to every *instance* of a trade item means that two otherwise identical units of the same product have distinct SGTINs. A product instance identified by an SGTIN is said to be *serialized*, and the process of assigning a unique SGTIN to a product and affixing a tag bearing that SGTIN in machine-readable form is called *serialization*.

Serialization makes it possible to trace individual products as they move through the supply chain. It also makes it possible to use Radio-Frequency Identification (RFID) to identify products – because RFID technology allows multiple tags to be read at once (unlike bar codes), distinct SGTINs are needed so that hardware and software can distinguish between reading two different tags versus reading the same tag twice. This guideline focuses especially on serialization when 96-bit RFID tags are to be used.

GS1 standards state the following:

- A product class is uniquely identified by a GTIN, whose structure is specified in the GS1 General Specifications. The GTIN is commonly encoded into a U.P.C. or EAN-13 bar code that is used at point of sale to determine what product is being purchased.
- A specific instance of a product class is uniquely identified by the combination of its GTIN and a serial number that is assigned uniquely to each instance. The serial number is unique within each GTIN; that is, it is acceptable to have a serial number 100 of Product A and a serial number 100 of Product B, but there ***cannot*** be two distinct instances of Product A that both have serial number 100. This is a critical to the success of tracing items at an instance level and when using RFID.
- GS1 standards define how the combination of GTIN + serial number may be encoded into bar code symbols, including 1-D bar codes (GS1-128, DataBar) and 2-D bar codes (GS1 Data Matrix, GS1 QR Code). See the GS1 General Specifications.
- GS1 standards also define how the combination of GTIN + serial number may be encoded into HF and UHF Gen 2 RFID tags. See the GS1 EPC Tag Data Standard. Certain RFID tags impose limitations on the value of the serial number, as noted below.
- GS1 standards specify that it is the responsibility of the product brand owner to assign a globally unique GTIN to each distinct product class, and to assign unique serial numbers to each product instance when product instances are to be serialized. The brand owner may delegate serialization to other parties, ***but it is still the brand owner's responsibility to ensure that serial numbers are assigned uniquely within each GTIN.*** See the GS1 General Specifications.

The purpose of this guideline is to supplement the GS1 Standards by saying *how* brand owners can manage the assignment of unique serial numbers to their products. In particular, this guideline offers several strategies that can be used when brand owners have to delegate the assignment of serial numbers to multiple parties, either internal divisions or manufacturing plants or external parties such as contract manufacturers and service bureaus.

The GS1 Standards specify that a serial number is an alphanumeric string of between 1–20 characters, drawn from a character set that includes digits, upper- and lowercase letters, and a variety of punctuation. However, 96-bit RFID tags (the most commonly available type, at present) do not have sufficient storage capacity to hold a GTIN plus a 20-character alphanumeric serial number. Therefore, when 96-bit RFID tags are used, serial numbers are restricted to be all numeric (i.e., the only characters permitted are the digits 0 through 9), and between 1-12 digits in length. (More precisely, the serial number must be all numeric, the first digit must not be a "0", and the value when

read as a decimal numeral must be less than or equal to 274877906943. Not all 12-digit numerals fit within this restriction.)

In this guideline, we focus on brand owners who intend to use 96-bit RFID tags as the primary data carrier for affixing the SGTIN to serialized products. We also focus on the serialization at the item level, though the principles discussed here would apply equally well at the case level or higher. We focus on business practices that are common in the apparel industry. While the principles of serialization apply equally to all industries, different industries may confront a different range of business issues. For example, in pharmaceuticals the serialization of product is *never* delegated to a third party as it is in apparel; conversely, in pharmaceuticals there are sometimes requirements for randomization of serial numbers which do not occur in apparel and are not discussed here.

The last section of this document discusses certain challenges of serialization that are not easily overcome given the current standards. That section discusses how future standards might provide new solutions that are not otherwise discussed in this document.

2. Scope

This guideline supplements GS1 Standards by offering several strategies for managing the assignment of unique serial numbers to their product in a standards-compliant way. This guideline is primarily aimed at brand owners operating within the following scope:

- Products in the apparel and fashion industry
- Products that already carry a GTIN (including U.P.C. or EAN-13) to identify the product class
- Products that are to be serialized at the item level
- Products that are to carry a 96-bit EPC RFID tag containing the GTIN plus a unique serial number
- Products that are mainly intended to be serialized “at source”; i.e., by the brand owner itself or a manufacturing or labelling partner of the brand owner, such that the product is serialized prior to delivery by the brand owner to its customer. There may be some limited need to assign serial numbers further downstream in the supply chain; e.g., in an exception situation where the source tag is missing.

The motivation for serializing the product is assumed to be one or more of the following:

- RFID tags are to be used for managing inventory. When RFID tags are used, each product must have a unique SGTIN so that hardware and software can distinguish between reading two different tags (on different product instances) and reading the same tag twice. Avoiding duplication is absolutely critical to realizing the benefits of RFID technology.
- Individual product instances are to be tracked or traced through the supply chain by correlating observations of the product instances obtained in different places at different times. The unique SGTIN makes it possible to correlate individual observations. Typically this involves sharing of data among two or more parties in the supply chain.

In principle, assigning unique serial numbers is a straightforward as using a counter 1, 2, 3, …, for each GTIN to assign the next unused serial number to each instance of that GTIN as it is manufactured. However, this becomes more complicated when serial numbers for the same product may be assigned in more than one physical location. Examples of situations in which this occurs are:

- The same product is manufactured on multiple manufacturing lines in the same building.
- The same product is manufactured in different manufacturing plants.
- One or more contract manufacturers are used
- Product labels (including RFID tags) are affixed to products by one or more 3rd party service bureaus

- Pre-programmed labels (including RFID tags) are obtained from one or more label service providers and affixed to the products later
- Downstream supply chain parties (e.g., retailers) need to assign new serial numbers in exception situations; e.g., where the original tag is missing, or a product returned by a customer with the label removed is to be restocked

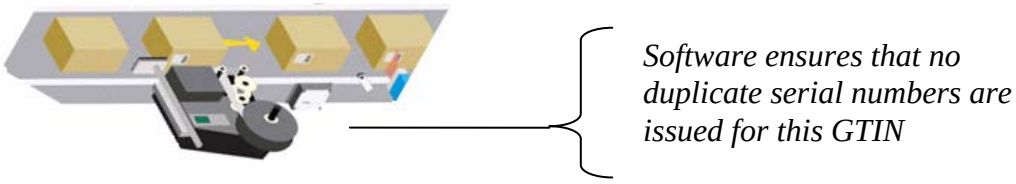
While this guideline focuses on the above scope, most of the principles of serialization are quite general and will find applicability in other industries and business settings as well.

3. Business Scenarios

This section illustrates commonly occurring business process scenarios for serialization of products. The next section illustrates different approaches for managing serial numbers to ensure uniqueness, and shows how they may be applied to each of the business scenarios in this section.

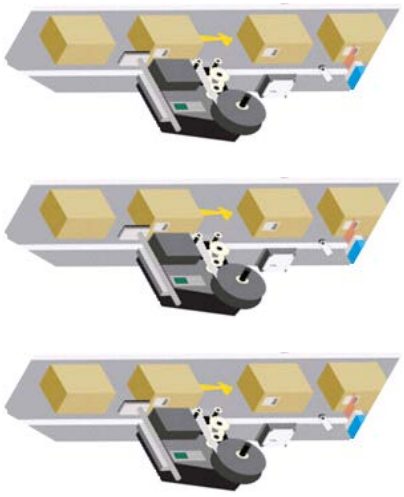
3.1. Brand Owner Manufacturing on a Single Line

The simplest business scenario for serializing product is when a brand owner manufacturers and tags a product on a single manufacturing line. Although this single manufacturing line is the only place where serial numbers for the given product are assigned, a process is needed to ensure uniqueness of serial numbers. The software that runs the manufacturing line has complete control over what serial numbers are issued; and so on its own it can employ a method to ensure uniqueness of serial numbers. This includes any of the methods discussed in Section 4.



3.2. Brand Owner Manufacturing on Multiple Lines

A more complex business scenario is where the same product (same GTIN) is manufactured and tagged on more than one manufacturing line. This includes having several manufacturing lines within the same plant, or manufacturing lines that are geographically distributed. In this case, the challenge is to ensure that one manufacturing line does not use the same serial number that a different manufacturing line has already used for the same product.



Each line ensures that no duplicate serial numbers are issued on that line. How does one line ensure it does not duplicate serial numbers issued by a different line?

3.3. Brand Owner Uses Service Bureaus, Contract Manufacturers, or Other 3rd parties

A brand owner may delegate serialization to 3rd parties in a variety of ways. For example, a brand owner may use a contract manufacturer who is responsible for manufacturing the product and tagging it. A brand owner may also use a tagging service bureau who is responsible for tagging finished products delivered to it by the brand owner, or who supplies pre-programmed tags that are affixed by the brand owner during in-house manufacturing.

In all of these scenarios, the unique serial number is assigned by a party other than the brand owner, though in all cases under the direction of the brand owner. This is similar to the scenarios in Sections 3.1 and 3.2, depending on whether the brand owner contracts with one or multiple 3rd parties, but with the additional complication that any steps the brand owner takes to ensure uniqueness of serial numbers must be done in conjunction with the 3rd parties.

3.4. Downstream (Distributor or Retailer) Exception Tagging

In all of the previous scenarios, the brand owner is ultimately responsible for serialization, either because the brand owner is doing the serialization or the brand owner has contracted to a 3rd party to do it under his direction. In some cases, however, a party further downstream in the supply chain may have a need to serialize a brand owner's product. Typically this party is a retailer, but it could also be a distributor or other intermediary. In this guideline, we have assumed that the majority of products are tagged "at source" (i.e., by the brand owner or under the brand owner's control), and so tagging by a downstream party only occurs in an exception situation. Such situations may include the following:

- A retailer (or other downstream party) receives a product with a missing or broken RFID tag, but wishes to create a new RFID tag so that the product can be handled alongside properly tagged products.
- A retailer wishes to restock a product that was returned by the customer, and the original RFID tag is no longer present.
- A retailer (or other downstream party) is in a state where its supplier has not yet begun or just begun source tagging, and the retailer wants to tag its existing untagged inventory rather than wait for the untagged inventory to be replaced by source-tagged inventory over time.

In some of these situations (e.g., a broken RFID tag) the retailer may be able to determine the original serial number (e.g., if the label still has legible human-readable text). In that case, the retailer could simply replace the tag with an identical copy. Otherwise, the retailer will have to assign brand-new identification to the product. Section 4.4 discusses some approaches for this case.

4. Approaches to Managing Serialization

This section provides best practice guidance for managing serialization, especially in situations where the same product is manufactured in multiple locations or by 3rd parties, or where the method of serialization is expected to change over time.

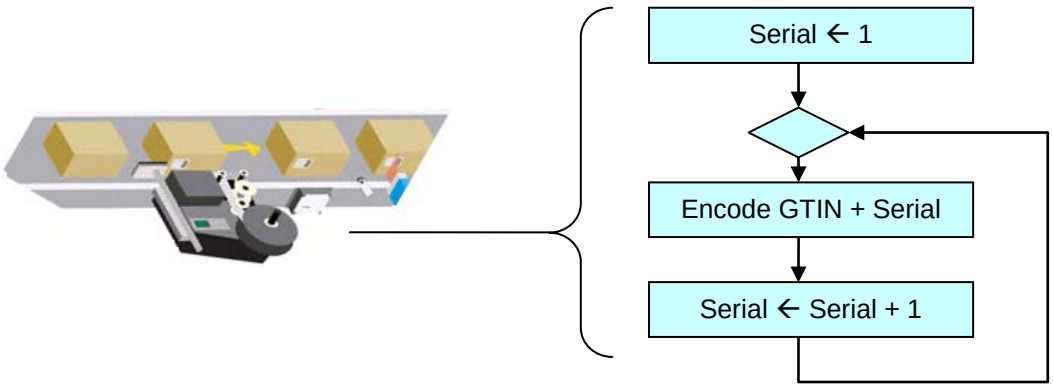
In these descriptions, it is assumed that 96-bit RFID tags are used. This limits the serial number to be all-numeric, with no leading zeros, so that the serial number is a decimal numeral in the range $0 \leq \text{serial} < 2^{38}$. That is, the lowest-numbered serial number is 0, and the highest-numbered serial number is 274877906943. When encoded into the RFID tag, the serial number is translated into 38 binary bits, where serial number 0 results in 38 zero bits and serial number 274877906943 results in 38 one bits. The serial number is generally written in decimal for purposes of human-readable representation, for encoding into bar codes (if used), in EDI messages such as Advanced Ship Notices, and in EPCIS data that is used to share supply chain visibility between trading partners. The binary equivalent only occurs in the RFID tag itself, and in certain low-level software that interfaces directly to RFID readers and printers. However, when managing the range of available serial numbers, it is sometimes convenient to think of the binary form of the serial number rather than decimal form. A familiarity with converting between the two is helpful.

4.1. IT-Based Serialization

The methods in this section are called “IT-based” because they rely upon the information systems of a tagging party to manage the allocation of serial numbers. The IT systems are used to keep track of which serial numbers have been allocated and which have not. The software performing this function can exist at a variety of levels within an enterprise’s IT architecture, from being embedded directly in a printer or other manufacturing device, to being a function of a Manufacturing Execution System (MES), to being a corporate-wide enterprise software function. Regardless, they all rely upon stored information that keeps track of which serial number to allocate next, and so it is important that these systems be properly secured and backed up.

4.1.1. Sequential Serialization on a Single Line

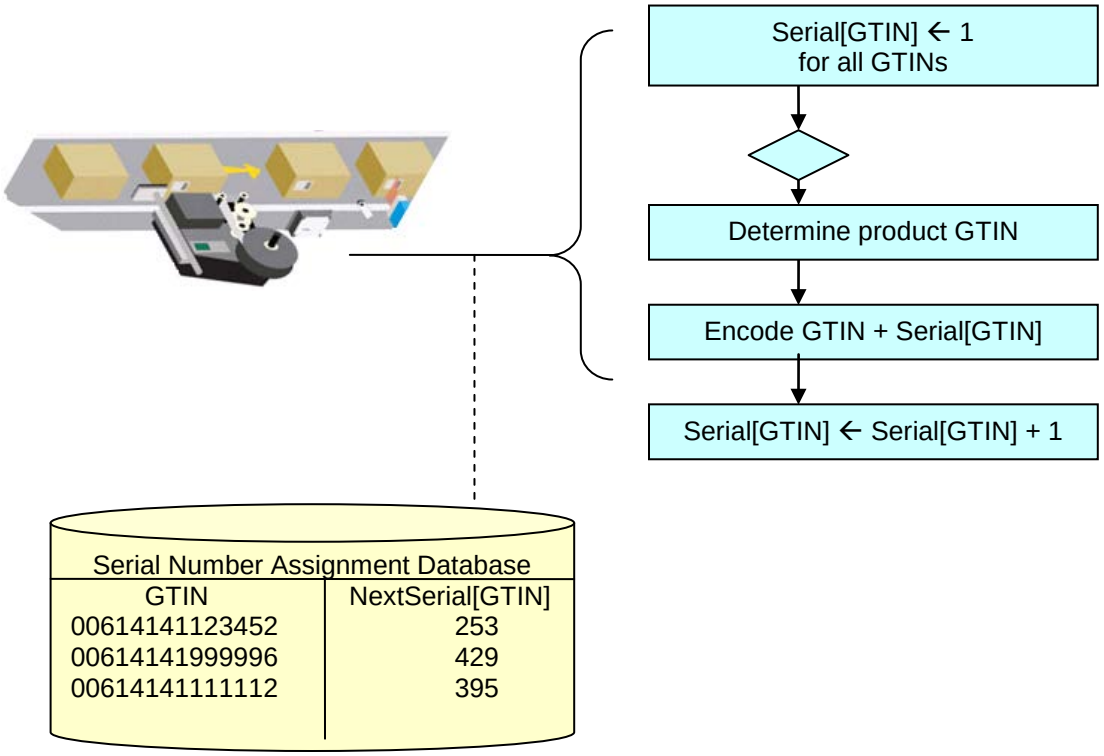
The basic building block for all IT-based serialization methods is the allocation of serial numbers on a single manufacturing or tagging line. The simplest method is to have a counter which allocate serial numbers one at a time, as illustrated below:



In this picture, the first instance of the product receives serial number 1, the second receives serial number 2, and so on. If the full 38-bit serial number available in the SGTIN-96 RFID tag is consumed in this way, there is capacity to serialize 2^{38} = 274,877,906,944 instances of a product (GTIN) without duplications.

The software responsible for sequential serialization need only keep track of a single number; namely, the next available serial number in the sequence. This is critical information to ensure that serial numbers are not duplicated.

The serial number only has to be unique within a given GTIN, and so when there are multiple products (multiple GTINs) there is a "next number" for each GTIN. When a single software system is responsible for serializing multiple GTINs, the picture looks something like this:



The serial number assignment database keeps track of the next available serial number for each GTIN being manufactured. As in the previous case, it is very important that the state information in this database be carefully backed up.

Typically, brand owners do not construct databases like this themselves, but rather this is a function that is built into the serialization systems obtained from solution providers. In the case where a brand owner works with a 3rd party such as a service bureau or contract manufacturer, the systems and databases are maintained by the 3rd party rather than the brand owner.

4.1.2. Static Allocation of Serial Number Ranges to Multiple Lines

When there are two or more manufacturing lines tagging the same product (same GTIN), they may each use sequential serialization as described above, but additional care must be taken to ensure that the two lines do not issue the same serial number. This is true whether the multiple manufacturing lines belong to the brand owner or to 3rd parties to whom the brand owner contracts (or even a combination of the two). In the illustrations that follow, "Line #1", "Line #2", and "Line #3" will be used

generically to illustrate three manufacturing lines, but they could also be different service bureaus, different contract manufacturers, and so forth.

A straightforward way to avoid duplication is to give each manufacturing line a separate set of serial numbers to use within each GTIN. In the "static allocation" approach, the entire range of possible serial numbers for a GTIN is divided into large blocks, each block assigned to a manufacturing line, and each manufacturing line allocates serial numbers for that GTIN within its specified block. There are many ways that dividing the serial number range into blocks can be done:

Contiguous Ranges in Decimal

Each manufacturing line can be given a contiguous range of serial numbers, expressed in decimal. Here is an example:

Manufacturing Line	Minimum Serial Number	Maximum Serial Number
Line #1	0	19999999
Line #2	20000000	39999999
Line #3	40000000	59999999

In this example, each manufacturing line is given a range of 20 million serial numbers to use. For example, Manufacturing Line #2 is free to assign any serial number, provided the number is greater than or equal to 20000000 and less than 40000000. If Manufacturing Line #2 is using sequential serialization, it simply initializes its counter to 20000000. It should also check to make sure the upper limit is not exceeded, though normally in static allocation the size of the range is well in excess of the number of products that could be possibly manufactured on any given line.

This scheme is simple to understand, gives flexibility add more manufacturing lines (because the entire serial number range has not been fully allocated), and the human-readable form of the serial number makes it easy to recognize which range was used.

Structured Serial Number in Decimal

A slightly different way to conceptualize assigning contiguous ranges of serial numbers is to think of building up the decimal serial number in pieces. For example, the following rule might be adopted:

- Each manufacturing line is assigned a two-digit code 10, 11, 12, etc.
- The serial number assigned on a given manufacturing line is composed of the line's two digit code, followed by seven digits assigned by the line.

The following table illustrates the ranges that result from the above rule:

Manufacturing Line	Line Code	Serial Number Pattern	Minimum Serial Number	Maximum Serial Number
Line #1	10	10nnnnnnn	100000000	109999999
Line #2	11	11nnnnnnn	110000000	119999999
Line #3	12	12nnnnnnn	120000000	129999999

As the table illustrates, the "structured serial number" approach is really no different than the "contiguous ranges" approach; the ranges have simply been defined in a different way. The structured serial number approach (in decimal) results in the size of each range being a power of ten; in the above example, each range contains 10 million serial numbers.

Contiguous Ranges and Structured Serial Numbers in Binary

In an EPC RFID tag, the serial number is encoded into a binary representation on the RFID chip, as a 38-bit unsigned binary numeral. Low-level RFID reading and writing software often works with the

binary form rather than the decimal form (often the binary form is shown in hexadecimal instead). Describing serial number ranges in binary format has the advantage that it is easier to describe the use of the full serial number range offered by the 96-bit RFID tag, because the upper limit expressed in binary is simply 38 "one" bits, whereas the decimal equivalent (274877906943) is not a "round" number. A structured serial number where the structure is described in binary may also be easier to manipulate by low-level software that is working in binary rather than decimal. The disadvantage of viewing serial numbers in binary is that it's not so easy to identify the range of a serial number when it is displayed in decimal or in bar code form.

Both the "contiguous range" and "structured serial number" approaches to defining ranges can be done on the basis of binary numbers. The result is equivalent to using the contiguous range approach in decimal, but the upper and lower limits expressed in decimal will not be "round" numbers.

Here is an example of a structured approach, expressed in binary:

- Each manufacturing line is assigned a three-bit binary code 000, 001, 010, etc.
- The serial number assigned on a given manufacturing line is composed of the line's 3-bit code, followed by 35 bits assigned by the line, resulting in a 38-bit serial number.

The following table illustrates the ranges that result from the above rule:

Manufacturing Line	Line Code	Serial Number Pattern (binary)	Minimum Serial Number (decimal)	Maximum Serial Number (decimal)
Line #1	000	000bbbbb…bbb	0	34359738367
Line #2	001	001bbbbb…bbb	34359738368	68719476735
Line #3	010	010bbbbb…bbb	68719476736	103079215103

In the "serial number pattern" column above, the actual serial number is 38 bits – in the table, several bits are omitted for reasons of space. The size of each range is 2^{35} = 34,359,738,368 serial numbers in each range.

Other Variations

There are many other ways to divide the entire range of serial numbers into different sets for different manufacturing lines. For example, if there are just two manufacturing lines, one could instruct Line #1 to only use even serial numbers and Line #2 to only use odd serial numbers. This is similar to having a 1-bit "line code" as illustrated above, but where the line code is the *least* significant bit of the binary serial number instead of the most significant bit. This idea can be extended to more than one bit; e.g., a 3-bit line code in the least significant position. One advantage of this approach is that the same structure will still be usable if in the future the manufacturer needs to go beyond the 38-bit serial number available in the 96-bit RFID tag.

Considerations for Using Static Allocation

The static allocation approach is very straightforward to apply. It also requires no special software. A record must be kept of what ranges have been allocated to what manufacturing lines, but because this allocation table is rather small and changes infrequently, even an informal spreadsheet or paper record may be adequate. A best practice would be to ensure that this record, whether in paper or electronic form, has adequate backup, and a succession plan for the person or group within the company that has stewardship of this record.

A challenge with the static allocation method is that the manufacturer to predict in advance both how many manufacturing lines will be needed for a product (i.e., how many ranges need to be allocated), and how many products of a given GTIN will be manufactured on those lines (i.e., how big to make each range). One way to address this challenge is to define many more ranges that are currently needed, and make each range as large as possible to fill the entire 38-bit serial number capacity. However, if later it is discovered that more ranges are needed, or more capacity within each range, it will be very difficult to adjust as previous allocations will need to be retracted to create additional

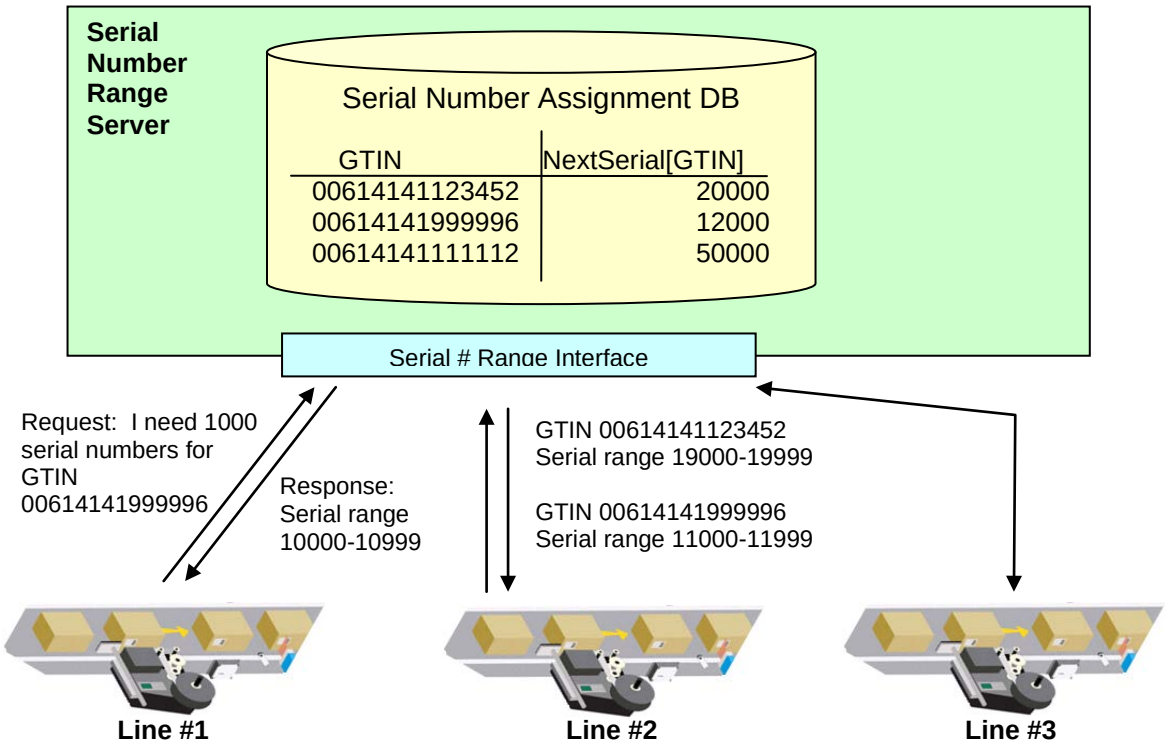
space. Alternatively, the initial static allocation can be set so that there is a large portion of the serial number space not allocated to any range. For example, the 38-bit serial number space could be divided in two pieces, and then serial numbers beginning (in binary) 0bbbb…. would be further divided into ranges for each manufacturing line, and serial numbers beginning 1bbbb…. would be reserved for future allocation. This gives flexibility in the future, at the expense of less available capacity for each range today.

Static allocation requires careful thought ahead of time in devising a suitable plan. Companies should consider the number of manufacturing lines for a given product both now and in the future (including contract manufacturers and service bureaus), and the likely volume of product both overall and on a per-line basis. The answers to these questions might not be the same for every product, and it is conceivable that different allocation plans could be used for different GTINs.

4.1.3. Dynamic Allocation of Serial Number Ranges to Multiple Lines

As noted above, the virtue of the static allocation method is very simple bookkeeping, with the disadvantage that the needs for serial numbers by different manufacturing lines must be predicted in advance. Dynamic allocation provides an alternative approach intended to mitigate the disadvantages of static allocation.

In dynamic allocation, ranges are allocated to manufacturing lines on a demand-driven basis, rather than in advance. This typically requires the deployment of a software system whose function is to assign ranges of serial numbers in response to requests. The following figure illustrates how this works:



The brand owner deploys a “serial number range server”, shown at the top of the figure, which is responsible for allocating blocks of serial numbers for each GTIN that the brand owner serializes. The serial number range server provides a network-based application programming interface (API) through which a manufacturing line can request a block of serial numbers. A manufacturing line issues a request, in which it specifies a GTIN, and a quantity of serial numbers it wants for that GTIN. The serial number range server allocates a block of unused serial numbers of the requested size, and responds to the manufacturing line by providing those serial numbers (e.g., by listing all the serial

numbers in the block, or providing the minimum and maximum numbers). The figure above illustrates Line #1 and Line #2 both requesting a block of 1000 serial numbers for GTIN 00614141999996. Line #1 receives serial numbers 10000 – 10999, while Line #2 receives serial numbers 11000 – 11999.

The serial number range server keeps track of the serial numbers allocated so that each request for a given GTIN is answered with a different range of numbers. The simplest implementation just tracks what the next available serial number is, as illustrated in the figure. The server might also keep a record of the blocks previously allocated and to which manufacturing line each was given.

While there is currently no standard defining the interface to a serial number range server (i.e., the API shown in the figure), there are several commercial software implementations available. These commercial implementations come with a proprietary API, but they are all roughly equivalent in the content of the request and response. If the brand owner wants to use dynamic range allocation with contract manufacturers, service bureaus, or other third parties, a provision must be made for the 3rd party to interface with the serial number range server maintained by the brand owner. Conversely, there are commercial offerings where a serial number range server is offered to brand owners via the Internet in a Software-as-a-Service (SaaS) mode. In that case, the brand owner's manufacturing lines interface to the SaaS service via the Internet to obtain serial number ranges.

4.2. Chip Based Serialization (TID)

All of the serialization methods discussed in Section 4.1 are based on the brand owner controlling its own serial number assignment through information systems it deploys. An alternative approach makes use of an RFID tag hardware feature called the Tag Identifier (TID). Because this method relies upon a hardware feature of the RFID tag, it is called "chip-based" serialization.

4.2.1. About the TID

The TID is a special memory within the RFID tag that holds information about the RFID tag itself, as opposed to information about the object to which the tag is affixed. All RFID tags include information in the TID that identifies the maker of the chip and the model (i.e., which of several different chip products the maker offers).[1] Many RFID tags also include additional information in the TID. One of these additional pieces of information is a serial number that is assigned by the manufacturer of the RFID chip – unique among all RFID chips of the same make and model. This serial number is referred to as the TID serial number.

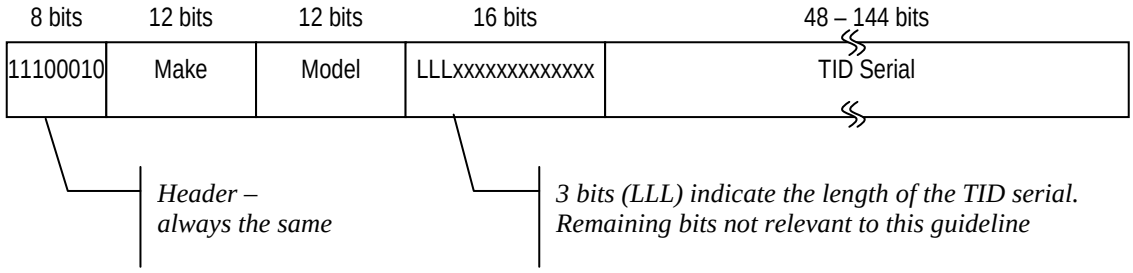
The TID serial number is different than the serial number that is part of the EPC. The EPC consists of a GTIN that identifies the product and a serial number assigned by the brand owner to identify a specific instance of that product. The serial number in the EPC is unique within a given GTIN. The TID serial number, in contrast, is assigned by the RFID chip manufacturer before the chip is affixed to a product (indeed, before the chip is even made into an RFID tag). The chip manufacturer has no idea to what product the chip will eventually be affixed, and so the TID serial number has nothing to do with the GTIN or EPC. The chip manufacturer simply changes the serial number for each chip it makes. If the chip make and model information from the TID is combined with the TID serial number, the result is a number that is different for *every* RFID tag manufactured by anybody.

[1] It is important to understand the difference between the RFID chip and an RFID tag. An RFID tag is what a brand owner affixes to a product. An RFID tag contains several parts. There is the paper or plastic label that attaches to the garment or other product. Embedded within the label is an RFID "inlay" that contains the RFID components. The inlay includes a tiny silicon RFID chip, less than a millimeter square, and a much larger metallic antenna. The RFID chip contains all of the electronic components that make the RFID tag work, including the radio receiver and the memory that holds the EPC and other information. RFID chips are typically made one company, then manufactured into a complete RFID tag by a different company. While there are many different RFID tag companies, there are relatively few RFID chip companies. The TID identifies the company that makes the RFID chip, *not* the maker of the RFID tag. This is because the TID is programmed at the time the chip is manufactured, before it is sold to the company who manufactures it into an RFID tag.

4.2.2. Using the TID to Serialize Products

To create a unique EPC for a product having a given GTIN, the brand owner needs a serial number that is different from every other serial number assigned for that GTIN. The TID serial (including the make and model) is different on every RFID chip. The idea in chip-based serialization is to leverage the TID serial number to create the EPC serial number. In other words, the brand owner creates an SGTIN by reading the TID serial from the RFID tag he is about to affix to the product, and combining the TID serial number with the GTIN to arrive at an SGTIN. This SGTIN will be different from every other SGTIN: the serial number by itself is unique, and so the combination of GTIN+serial is clearly unique as well. For a given GTIN, the serial numbers used will appear to be fairly random as compared to sequential allocation – there will be many "holes" in the serial number range for a given GTIN, corresponding to serial numbers that were used for other GTINs. But the overall serial number range is large enough to provide sufficient capacity even if there are many such "holes."

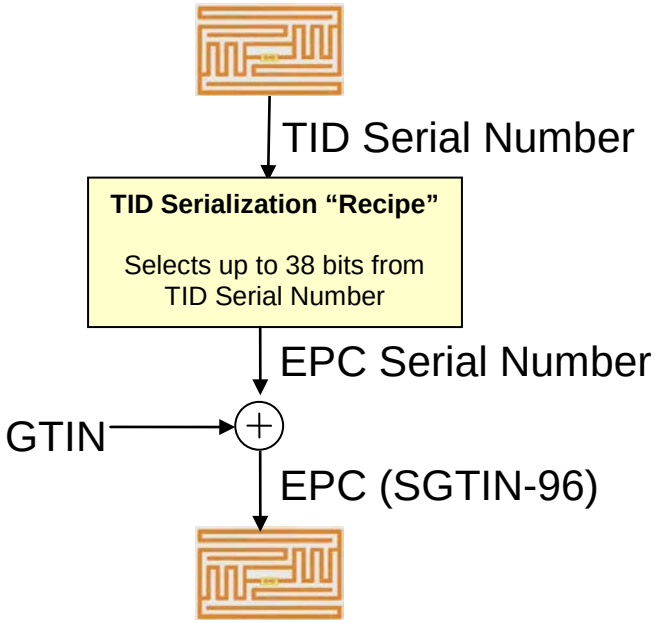
This idea is not as straightforward as it appears, however, because the TID serial number as specified in the EPC Tag Data Standard does not fit into the EPC serial number. On a 96-bit RFID tag, the serial number in the EPC is 38 bits, giving a capacity for 2^{38} = 274,877,906,944 different serial numbers. The TID as specified in the EPC Tag Data Standard, however, has a much more complex format, illustrated below:



The first eight bits of the TID are a header that is the same for all EPC RFID tags. The next 12 bits indicate the make of the chip, and the 12 bits after that indicate the model. Following that are 16 more bits, three of which indicate the length of the TID serial number (the other 12 bits in this segment are not relevant to this guideline). After that comes the TID serial number itself, which can be 48, 64, 80, 96, 112, 128, or 144 bits in length. Recalling that the chip make and model must be combined with the TID serial in order to arrive at a number that is unique across all RFID chips, this implies a minimum of 12+12+48 = 72 bits must be extracted from the TID to make a unique serial number, and possibly as many as 12+12+144 = 168 bits, depending on how many bits the chip manufacturer has chosen to use in its TID.

Regardless of the size of the TID, the number of bits that must be extracted to have a guaranteed unique serial number across all RFID chips is many more than the 38 bits that are available in the serial portion of the EPC. Blindly taking 38 bits out of the TID is not sufficient to avoid duplicate serial numbers because two RFID chips might have TIDs that are the same in those chosen 38 bits. On the other hand, if the brand owner knew how the chip manufacturer was assigning its TID serial numbers, the brand owner could predict whether duplication is a risk. For example, if the brand owner knows the chip manufacturer simply starts at TID serial number 1 and increments by one for each chip it makes, then with that knowledge the brand owner could extract the least significant 38 bits of the TID serial number. Many of the available serial numbers won't be used for the brand owner's products – they are spread across the total production by that chip manufacturer. The brand owner must work with the chip manufacturer to understand the period of time over which duplication becomes a risk. Not all chip manufacturers assign TID serial numbers sequentially as in the above example, and so the choice of which TID bits to use must also be guided by the chip manufacturer.

With this in mind, chip-based serialization works like this:



When the brand owner programs an RFID tag to put on a product, he first reads the contents of the RFID tag's TID memory. The brand owner then applies a "formula" for extracting some number of bits out of the TID memory contents – possibly a full 38 bits, possibly fewer as discussed below – and then uses the result of the formula together with the GTIN to create the EPC.

The "formula" for extracting the right bits from the TID has to be constructed in a way that fits with the way the chip manufacturer assigns TID serial numbers. Because chip manufacturers generally do not reveal their method for assigning TID serial numbers, the brand owner must rely upon the chip manufacturer to provide a suitable formula, tied to the particular make and model of chip the brand owner uses. The chip manufacturer must provide three things to the brand owner:

- The formula itself – that is, exactly what bits of the TID should be extracted, and how they should be manipulated to arrive at the bits that will go into the EPC serial number.
- How many bits the formula yields. This might be a full 38 bits, or it might be fewer as discussed below.
- Over what period of time does the chip manufacturer expect the formula to yield unique results (i.e., no duplicate serial numbers), and is this an absolute guarantee or is it just a high probability. This is partly a function of how the chip manufacturer assigns the serial numbers, and partly a function of how many chips using the same formula the chip manufacturer expects to make over time.

With this information, the brand owner will know how to assign serial numbers, and what to expect in terms of uniqueness.

Note that the application of the formula is performed by the software that controls the device that is programming the RFID tags. Often, the formula is built into a device such as an RFID printer, and so the details of how the formula works is arranged between the chip manufacturer and the printer vendor. The brand owner only needs to know how many bits the formula uses (for reasons discussed below) and the uniqueness properties promised by the chip manufacturer for that formula.

The formula has to be coordinated with the chip manufacturer's method for assigning TID serial numbers. It is important to have a clear understanding of what chips a formula applies to, and what the uniqueness guarantees are.

4.2.3. Chip-based Serialization on Multiple Manufacturing Lines

In Section 4.1, IT-based serialization using simple sequential assignment was introduced. When this process was extended to multiple manufacturing lines, it was necessary for the brand owner to use some means to coordinate the ranges used by the different lines. Static allocation and dynamic allocation were introduced as two methods for doing this.

With chip-based serialization, multiple manufacturing lines can be accommodated without any such coordination. Because the TID serial number is unique across *all* RFID chips, any chip-based serial number will be different from every other, even if it is assigned on a different manufacturing line. So to accommodate multiple manufacturing lines, each line uses the appropriate formula(s) for the chips used on that line. There is no pre-allocation of ranges or bookkeeping as in static range allocation (Section 4.1.2), nor any serial number range server as in dynamic range allocation (Section 4.1.3). Adding a new manufacturing line for the same product requires no extra work – no new ranges to allocate, and no new integration with a serial number range server.

It is important to note, however, that a chip-based serialization approach does not preclude a brand-owner from using an IT infrastructure for recording the serial numbers that have been applied to product variable data, data sharing with trading partners, and other purposes.

4.2.4. Considerations for Using Chip-based Serialization

When using chip based serialization, exactly which serial numbers are used is unpredictable from the brand owner's standpoint. TID serial numbers are not necessarily assigned sequentially by the chip manufacturer. Even if the chip manufacturer assigns sequential TID serial numbers, the EPC serial numbers for a given GTIN will not appear to be sequential, because only a fraction of chips will be affixed to instances of that GTIN. All the brand owner knows is that the serial numbers yielded by the TID serialization formula will be unique, within the uniqueness guarantee for the formula provided by the chip manufacturer.

Deriving all 38 bits of the GTIN serial number from the TID presents difficulties if the brand owner wants to use more than one serialization method, or wants to switch to a different chip manufacturer (implying a different formula), or wants to switch from chip-based serialization to some other method. Let us illustrate how the difficulty arises. Consider a brand owner who is using IT-based serialization, assigning his own numbers sequentially. After manufacturing one million products, the brand owner decides to switch to dynamic range allocation. The brand owner knows that the only serial numbers he has allocated fall within the range 1 – 1,000,000. The brand owner switches to dynamic range allocation by initializing his serial number range server to begin issuing blocks at serial number 1,000,001.

Now consider a similar situation where a brand owner has been using chip-based serialization, where the formula yields the full 38 bits of the serial number. After manufacturing one million products, there are still a huge number of available serial numbers – out of 274+ billion available serial numbers, over 99.9996% are still available for use. However, the remaining numbers are not in one contiguous range. Even if the brand owner kept a record of what serial numbers were previously used, they will be scattered randomly throughout the 274+ billion possible numbers. It is not necessarily straightforward to find usable ranges that could be applied to other serialization methods, especially if a structured method is to be used. A similar problem occurs if the brand owner wants continue to use chip-based serialization but switch to a different chip manufacturer having a different formula – there is no guarantee that the new formula will yield numbers that are different from the old numbers.

A best practice for chip-based serialization, therefore, is to take steps to protect against this kind of change. The way this is done is to use a chip-based formula that yields *fewer* than 38 bits. To illustrate how this works, here is an example serialization plan:

- The brand owner decides to use chip-based serialization, using a formula that yields 36 bits of serial number from the TID.

- To create the 38-bit EPC serial number for a given tag, combine two zero bits (00) with the 36 bits yielded by the TID serialization formula, for a total of 38 bits. In this way, all serial numbers will begin with two zero bits.
- Serial numbers beginning with 01, 10, and 11 are reserved by the brand owner for future use.

The brand owner now has the flexibility to change methods in the future by using one of the reserved prefixes. For example, if the brand owner switches to a different chip manufacturer, he would ask that chip manufacturer for a 36-bit formula, and when using the new chips create the serial number by combining the bits 01 with the 36 bits from the new formula. Some of the 36-bit patterns yielded from the new formula may be duplicates of those yielded by the old formula, but the overall serial numbers will not be duplicates because the first two bits will be different. Likewise, the brand owner could switch to IT-based serialization, using a serial number range that begins with one of the reserved prefixes.

This is why the definition of a TID serialization formula provided for the possibility of the formula yielding fewer than 38 bits. Now in general, a formula yielding fewer bits will have a shorter interval of time before duplication may occur, because there are fewer possible serial numbers in fewer bits. For example, a chip manufacturer may offer a 38-bit formula that has a 10-year guarantee of no rollover, and a 37-bit formula that only has a 5-year guarantee. So the brand owner has a trade-off: longer guarantee with less flexibility to change, or vice versa.

The idea of using prefixes to allow for changing from one serialization method to another can be generalized to allow for several different methods simultaneously. This is discussed in the next section.

4.3. Creating a Top-level Serialization Plan

Sections 4.1 and 4.2 outline IT-based and chip-based approaches that a brand owner can employ to manage serialization. These approaches give a brand owner a wide variety of options that can be tailored to meet that brand owner's particular business requirements.

A best practice for brand owners is to create a top-level plan to define what serialization approaches are to be used. A good top-level plan clearly specifies what approach is to be used, drawing from the variety of approaches discussed in this guideline. In constructing a top-level plan, the brand owner should gather the following information:

- What products are to be serialized? What are their GTINs?
- What is the expected volume of each GTIN that will be manufactured over the life of the GTIN? This helps to assess how many serial numbers will be needed over time.
- Where will serialization take place? In the brand owner's own manufacturing facility, in 3rd parties contracted by the brand owner (contract manufacturers, serviced bureaus, etc), by other supply chain parties?
- How many different internal facilities and/or 3rd parties will be used?
- What IT capabilities are available, or can be made available, to manage serialization?
- How are the answers to the above expected to change over time?

These questions will help select among the various IT-based and chip-based approaches outlined earlier.

The conclusion from analyzing the available approaches is often that more than one approach needs to be supported. Sometimes, the brand owner sees a need to use two approaches simultaneously; e.g., if a given product is manufactured both in-house and by contract manufacturers, the brand owner may make a large static allocation to the contract manufacturers, and use dynamic allocation to manage multiple in-house manufacturing lines. Or, a brand owner needs to use two different chip manufacturers simultaneously, and each provides a different chip-based serialization formula. Other

times, only one method is to be used at any given time, but the brand owner wants flexibility to change to a different method in the future; e.g., the brand owner may use chip-based serialization using chip manufacturer A today, but wants the flexibility to switch to chip manufacturer B or even IT-based serialization in the future.

This means that a top-level plan will generally include a high-level allocation of the full 38-bit serial number space, so that these different choices can be accommodated. Here is an example to illustrate the concept. Brand Owner XYZ needs to meet the following requirements for its ABC product:

- Some units of ABC are manufactured in-house. XYZ has 10 manufacturing lines that might be used for this purpose, and expects to add 10 more lines over the next 20 years. Each manufacturing line is capable of managing its own serial numbers using sequential allocation.
- Some units of ABC are manufactured by contract manufacturers. This is only done at times of peak demand or when in-house facilities are unavailable, so the volume is relatively small compared to in-house manufacture. XYZ would like the contract manufacturers to use chip-based serialization to minimize the amount of bookkeeping required to manage serialization in this case. All of XYZ's contract manufacturers use RFID chips made by the Acme RFID Chip Co.
- XYZ recognizes that its industry is changing, and that its approach to manufacturing may change over the next 10 years. It wants the flexibility to alter its decisions regarding serialization in the future.

With these goals in mind, here is what a top-level serialization plan might look like for XYZ's ABC product:

Serial Number Range (binary)	Intended Use	Capacity
00ppppppsss…sss	In-house manufacturing. This is further divided into ranges for each manufacturing line, with the six bits (pppppp) assigned statically to each line, and the remaining 30-bit unique number generated on each line. Current assignments for pppppp are: 000000 = Boston line #1 000001 = Boston line #2 000010 = Boston line #3 000011 = Chicago line #1 000100 = Chicago line #2 000101 = Rotterdam line #1 000110 = Oslo line #1 000111 = Oslo line #2 001000 = Boston line #4 001001 = Reserved for RFID lab	Up to 64 different manufacturing lines, each with a capacity to issue 230 = 1,073,741,824 unique serial numbers.
01ccbbbbbb…bbb	Reserved for chip-based serialization. For chips made by Acme RFID Chip Co, the bits cc are set to 00. Acme must provide a 34-bit formula to complete the serial number. Other values of cc (01, 10, and 11) are reserved for future use; e.g., in case a different chip manufacturer is selected.	Up to four different chip-based serialization formulas over time.

Serial Number Range (binary)	Intended Use	Capacity
1bbbbbbbbbbbbb	Reserved. All serial numbers whose most significant bit is “1” are reserved for future use.	Half of all possible serial numbers for 96-bit tags are reserved for future use, a total of 2^{37} = 137,438,953,472 reserved numbers.

In this table, the 38-bit serial number is divided into three ranges, the first two each being a quarter of the total range, and the remaining half reserved for future use. The first range is allocated for sequential serialization by in-house manufacturing lines, and this range is further subdivided by static allocation to provide an individual range (approximately 1 billion serial numbers) for each line. There is room for up to 64 manufacturing lines, of which 10 are used today. The second range is allocated for chip-based serialization by third parties, of which a quarter has been designated for Acme tags using a 34-bit formula. In this way, there is adequate space for all current serialization methods, and room for change in the future.

In most cases, a serialization plan will not be as complex as depicted above. Some brand owners may simply divide the available space in two, using one half for a single method that is in use today, and the remaining half reserved for future use. Or, a brand owner may choose to commit to chip-based serialization forever, and may choose a chip-based serialization formula that yields all 38 bits. In general, the larger the brand owner, the higher volume the product, and/or the more complex the manufacturing arrangement, the more complex the top-level serialization plan will be.

Regardless of the methods chosen in the top-level plan, the brand owner is still responsible for quality control of serialization, including ensuring that the correct GTIN and serial number according to the plan is encoded into the tag, that the tag is readable, and that verification is performed to ensure serial numbers are not duplicated.

4.4. Supporting Downstream Exception Serialization

All of the preceding discussion pertains to serialization “at source” – by the brand owner directly or by a third party contracted by the brand owner. There are some situations, however, where serialization may be required downstream in the supply chain from the brand owner; for example, where the original tag is missing and replaced by a distributor or retailer, a product returned by a customer to a retailer with the label removed is to be restocked, and so on. These are referred to here as “exception tagging” scenarios, as they are assumed to be relatively infrequent compared to the volume of source tagged product. In such exception situations, the brand owner does not have direct control over the serial number being created – it is chosen by a downstream party. This introduces additional challenges not considered in the previous sections.

In principle, any of the previously discussed approaches could be extended to accommodate serialization by a downstream party. However, there are practical problems which may make these methods awkward or infeasible at the present time for some supply chain participants. In particular:

- *Static Allocation* In principle, a brand owner can reserve a range of the serial number space for use by its downstream trading partners, and allocate a separate subrange for each distributor and retailer who might need to generate an exception serial number. However, this can create challenges for some brand owners in communicating ranges to all downstream trading partners who might need to create exception tags. It may also be challenging for the downstream distributors or retailers who may need to manage different subranges provided by a multitude of suppliers.
- *Dynamic Allocation* In principle, a brand owner could deploy a serial number management server to allocate ranges, and make this server available via the Internet for use by downstream trading partners. However, there are currently no standard message format for

serial number range management, and so distributors and retailers might have to manage different proprietary interfaces from multiple suppliers.

- *Chip-based Serialization* In principle, downstream parties could use chip-based serialization to obtain unique serial numbers without any prior coordination with the brand owners. However, this requires that the retailer use the same top-level serialization plan as the brand owner, which can constrain the retailer's choice of chip manufacturer depending on the brand owner's top-level plan. As different brand owners may make different choices in this regard, again the distributor or retailer might have to do something different for each brand owner.

These methods may work effectively for certain brand owners and their downstream trading partners, but because of the limitations it is not possible at this time to provide a single rule that *all* downstream parties to follow to serialize products from *any* brand owner. Retailers may work out bi-lateral arrangements with individual brand owners who make provision for the retailer in their top-level serialization plans.

Because of this, at the present time some retailers avoid attempting to create a serialized GTIN to identify products that require exception tags. Instead, such retailers will affix some other sort of identification to such products. Several approaches to this are in common use:

- The retailer will program an EPC using an internal code; e.g., an SGTIN based on a retailer-created GTIN. The association of this internal code to the product's actual GTIN is made in a database the retailer uses for its own business processes.
- The retailer will use a binary encoding of the tag that goes outside of the EPC Tag Data Standard, resulting in RFID tag contents that are different from any source-tagged product (which all carry legitimate EPCs), but from which the product GTIN can be recovered according to a retailer's proprietary scheme.
- The retailer will program an RFID tag using a Global Individual Asset Identifier (GIAI), providing a globally unique identifier but one which is not related to the GTIN of the product. The retailer may associate the GIAI to the product GTIN in a database the retailer uses for its own business processes, or embed the GTIN into the GIAI serial number in a proprietary way.

All of these approaches allow the retailer to ensure that exception tags will not duplicate each other or proper SGTIN tags created at source. However, because exception tags created in this way do not carry proper SGTINs, it creates difficulties for sharing data about exception-tagged products across the supply chain.

GS1 will continue to work with industry to ensure its requirements are met over time. This includes the possibility of future standards developments to provide additional options for SGTIN-based exception tagging, suitable for use in an open supply chain where any downstream party might create exception tags for products from any brand owner.

THE GLOBAL LANGUAGE OF BUSINESS

CORPORATE HEADQUARTERS
Princeton Pike Corporate Center
1009 Lenox Drive, Suite 202,
Lawrenceville, New Jersey 08648 USA
T +1 937.435.3870 E info@gs1us.org
www.gs1us.org

twitter LinkedIn YouTube