



1

2

3

4

5

6

7

8

**EPC™ Generation 1 Tag Data Standards Version  
1.1 Rev.1.27**

9

10

11

12

Standard Specification  
10 May 2005

13

14

15

16

17

18

19

20

21

22

23

24

25  
26  
27

## DOCUMENT HISTORY

<b>Document Number:</b>	1.1
<b>Document Version:</b>	1.27
<b>Document Date :</b>	2005-05-10

28  
29  
30  
31  
32

## Document Summary

<b>Document Title:</b>	<b>EPC™ Generation 1 Tag Data Standards Version</b>
<b>Owner:</b>	Tag Data Standard Work Group
<b>Status:</b>	( <i>check one box</i> ) <input type="checkbox"/> DRAFT <input checked="" type="checkbox"/> Approved

33  
34  
35

## Document Change History

<b>Date of Change</b>	<b>Version</b>	<b>Reason for Change</b>	<b>Summary of Change</b>
03-31-2004	1.24	Update errata	<ul style="list-style-type: none"> <li>• Comments and errata identified during public review</li> </ul>
08-12-2004	1.25	Update errata	<ul style="list-style-type: none"> <li>• Further errata identified after release of v1.24 – especially inconsistencies regarding SSCC-96 and GRAI-96 partition tables</li> </ul>
09-16-2004	1.25	Update errata	<ul style="list-style-type: none"> <li>• Correct errors re numeric range of Asset Type of GRAI-96 and include further reference to Appendix F in section 5 – i.e. need to check valid numeric ranges for 64-bit and 96-bit tags</li> </ul>
11-19-2004	1.26	Update errata	<ul style="list-style-type: none"> <li>• Added clarification to restrictions to serial numbers in SGTIN, GRAI, GIAI</li> </ul>
05-10-2005	1.27	Revision	<ul style="list-style-type: none"> <li>• Added DoD construct header</li> <li>• Added hexadecimal expression for raw URI representation</li> <li>• Added disclaimer regarding non-applicability of TDS ver 1.1 to Gen 2 tags</li> <li>• Changed doc title to indicate tag generation covered by this doc and to quickly differentiate between the doc and the next version. Changes in the definition of the Filter Values (Section 3.4.1, 3.4.2, 3.5.1, 3.5.2)</li> <li>• Changes in the Filter Value tables (Table 5, 9, 13, 17, 21)</li> <li>• Changes in the Filter Value table in Appendix A, “Encoding Scheme Summary Table”</li> <li>• Addition of word “Indicator Digit” in encoding</li> </ul>

			process step 3 for SGTIN-64(Section 3.4.1.1) and SGTIN-96(Section 3.4.2.1) <ul style="list-style-type: none"><li>• Addition of word "Extension Digit" in encoding process step 3 for SSCC-64(Section 3.5.1.1) and SSCC-96(Section 3.5.2.1)</li></ul>
--	--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

36

## 37 **Abstract**

38 This document defines the EPC Tag Data Standards version 1.1. These standards define  
39 completely that portion of EPC tag data that is standardized, including how that data is  
40 encoded on the EPC tag itself (i.e. the EPC Tag Encodings), as well as how it is encoded  
41 for use in the information systems layers of the EPC Systems Network (i.e. the EPC URI  
42 or Uniform Resource Identifier Encodings). Readers should be advised that this Tag  
43 Data Specification Version 1.1 only applies to tag types in common use at the time of its  
44 publication. In particular, it does not provide specific guidance for using UHF Class 1  
45 Generation 2 tags ("Gen 2 tags"). It is intended that future Tag Data Specification will  
46 add guidance for use of Gen 2 tags, along with any substantive changes to the Tag Data  
47 Specification needed to support aspects of Gen 2 tags that differ from earlier tag types.

48  
49 The EPC Tag Encodings include a Header field followed by one or more Value Fields.  
50 The Header field defines the overall length and format of the Values Fields. The Value  
51 Fields contain a unique EPC Identifier and optional Filter Value when the latter is judged  
52 to be important to encode on the tag itself.

53 The EPC URI Encodings provide the means for applications software to process EPC  
54 Tag Encodings either literally (i.e. at the bit level) or at various levels of semantic  
55 abstraction that is independent of the tag variations. This document defines four  
56 categories of URI:

- 57 1. URIs for pure identities, sometimes called "canonical forms." These contain only  
58 the unique information that identifies a specific physical object, and are  
59 independent of tag encodings.
- 60 2. URIs that represent specific tag encodings. These are used in software  
61 applications where the encoding scheme is relevant, as when commanding  
62 software to write a tag.
- 63 3. URIs that represent patterns, or sets of EPCs. These are used when instructing  
64 software how to filter tag data.
- 65 4. URIs that represent raw tag information, generally used only for error reporting  
66 purposes.

67

## 68 **Status of this document**

69 This section describes the status of this document at the time of its publication. Other  
70 documents may supersede this document. The latest status of this document series is  
71 maintained at EPCglobal. This document is the ratified specification named Tag Data  
72 Standards Version 1.1 Rev.1.27. Comments on this document should be sent to  
73 [epcinfo@epcglobalinc.org](mailto:epcinfo@epcglobalinc.org).

## 74 **Changes from Previous Versions**

75 Version 1.1, as the first formally specified version, serves as the basis for assignment and  
76 use of EPC numbers in standard, open systems applications. Previous versions, consisting  
77 of technical reports and working drafts, recommended certain headers, tag lengths, and  
78 EPC data structures. Many of these constructs have been modified in the development of  
79 Version 1.1, and are generally not preserved for standard usage. Specifically, Version 1.1  
80 supersedes all previous definitions of EPC Tag Data Standards.

81

82 Beyond the new content in Version 1.1 (such as the addition of new coding formats), the  
83 most significant changes to prior versions include the following:

- 84 1. Redefinition and clarification of the rules for assigning Header values: (i) to allow  
85 various Header lengths for a given length tag, to support more encoding options in  
86 a given length tag; and (ii) to indicate the tag length via the left-most  
87 (“preamble”) portion of the Header, to support maximum reader efficiency.
- 88 2. Withdrawal of the 64-bit Universal Identifier format Types I-III, previously  
89 identified by specific 2-bit Headers. The Header assigned to the previous  
90 Universal Type II is now assigned to the 64-bit SGTIN encoding. The Type I and  
91 III Headers have not been reassigned to other encodings, but are rather simply  
92 designated as “reserved.” The Headers associated with Types I and III will  
93 remain reserved for a yet-to-be-determined period of time to support tags that  
94 have previously used them, unless a clear need for them arises (as was the case  
95 with the SGTIN), in which case they will be considered for reassignment.
- 96 3. Renumbering of the 96-bit Universal Identifier Header to fit within the revised  
97 Header rules, and renaming this code the “General Identifier” to avoid confusion  
98 with the Unique Identifier (UID) that will be introduced by the US Department of  
99 Defense and its suppliers.
- 100 4. Addition of DoD construct headers and URI expression.
- 101 5. Addition of hexadecimal expression for raw URI representation.

102

103 **Table of Contents**

104 1 Introduction ..... 9

105 2 Identity Concepts..... 10

106 2.1 Pure Identities ..... 11

107 2.1.1 General Types ..... 11

108 2.1.2 EAN.UCC System Identity Types ..... 12

109 2.1.2.1 Serialized Global Trade Item Number (SGTIN)..... 13

110 2.1.2.2 Serial Shipping Container Code (SSCC) ..... 14

111 2.1.2.3 Serialized Global Location Number (SGLN)..... 15

112 2.1.2.4 Global Returnable Asset Identifier (GRAI) ..... 16

113 2.1.2.5 Global Individual Asset Identifier (GIAI)..... 16

114 2.1.3 DoD Identity Type ..... 17

115 3 EPC Tag Bit-level Encodings ..... 17

116 3.1 Headers ..... 18

117 3.2 Notational Conventions ..... 20

118 3.3 General Identifier (GID-96)..... 21

119 3.3.1.1 GID-96 Encoding Procedure ..... 22

120 3.3.1.2 GID-96 Decoding Procedure..... 22

121 3.4 Serialized Global Trade Item Number (SGTIN) ..... 23

122 3.4.1 SGTIN-64 ..... 23

123 3.4.1.1 SGTIN-64 Encoding Procedure ..... 25

124 3.4.1.2 SGTIN-64 Decoding Procedure ..... 25

125 3.4.2 SGTIN-96 ..... 26

126 3.4.2.1 SGTIN-96 Encoding Procedure ..... 28

127 3.4.2.2 SGTIN-96 Decoding Procedure ..... 29

128 3.5 Serial Shipping Container Code (SSCC)..... 30

129 3.5.1 SSCC-64 ..... 30

130 3.5.1.1 SSCC-64 Encoding Procedure ..... 32

131 3.5.1.2 SSCC-64 Decoding Procedure ..... 32

132 3.5.2 SSCC-96 ..... 33

133 3.5.2.1 SSCC-96 Encoding Procedure ..... 34

134	3.5.2.2	SSCC-96 Decoding Procedure .....	35
135	3.6	Serialized Global Location Number (SGLN) .....	36
136	3.6.1	SGLN-64 .....	36
137	3.6.1.1	SGLN-64 Encoding Procedure .....	37
138	3.6.1.2	SGLN-64 Decoding Procedure .....	38
139	3.6.2	SGLN-96 .....	39
140	3.6.2.1	SGLN-96 Encoding Procedure .....	40
141	3.6.2.2	SGLN-96 Decoding Procedure .....	41
142	3.7	Global Returnable Asset Identifier (GRAI) .....	41
143	3.7.1	GRAI-64 .....	42
144	3.7.1.1	GRAI-64 Encoding Procedure .....	43
145	3.7.1.2	GRAI-64 Decoding Procedure .....	44
146	3.7.2	GRAI-96 .....	44
147	3.7.2.1	GRAI-96 Encoding Procedure .....	46
148	3.7.2.2	GRAI-96 Decoding Procedure .....	46
149	3.8	Global Individual Asset Identifier (GIAI) .....	47
150	3.8.1	GIAI-64 .....	48
151	3.8.1.1	GIAI-64 Encoding Procedure .....	49
152	3.8.1.2	GIAI-64 Decoding Procedure .....	49
153	3.8.2	GIAI-96 .....	50
154	3.8.2.1	GIAI-96 Encoding Procedure .....	51
155	3.8.2.2	GIAI-96 Decoding Procedure .....	52
156	3.9	DoD Tag Data Constructs (non-normative) .....	53
157	3.9.1	DoD-64 .....	53
158	3.9.2	DoD-96 .....	53
159	4	URI Representation .....	54
160	4.1	URI Forms for Pure Identities .....	54
161	4.2	URI Forms for Related Data Types .....	57
162	4.2.1	URIs for EPC Tags .....	57
163	4.2.2	URIs for Raw Bit Strings Arising From Invalid Tags .....	58
164	4.2.3	URIs for EPC Patterns .....	59
165	4.3	Syntax .....	60

166	4.3.1	Common Grammar Elements .....	60
167	4.3.2	EPCGID-URI.....	61
168	4.3.3	SGTIN-URI.....	61
169	4.3.4	SSCC-URI.....	61
170	4.3.5	SGLN-URI.....	61
171	4.3.6	GRAI-URI.....	61
172	4.3.7	GIAI-URI.....	62
173	4.3.8	EPC Tag URI.....	62
174	4.3.9	Raw Tag URI.....	63
175	4.3.10	EPC Pattern URI.....	63
176	4.3.11	DoD Construct URI.....	63
177	4.3.12	Summary (non-normative) .....	64
178	5	Translation between EPC-URI and Other EPC Representations .....	66
179	6	Semantics of EPC Pattern URIs.....	73
180	7	Background Information .....	74
181	8	References .....	74
182	9	Appendix A: Encoding Scheme Summary Tables.....	75
183	10	Appendix B: EPC Header Values and Tag Identity Lengths.....	80
184	11	Appendix C: Example of a Specific Trade Item (SGTIN) .....	82
185	12	Appendix D: Decimal values of powers of 2 Table.....	85
186	13	Appendix E: List of Abbreviations.....	86
187	14	Appendix F: General EAN.UCC Specifications.....	87
188			
189			

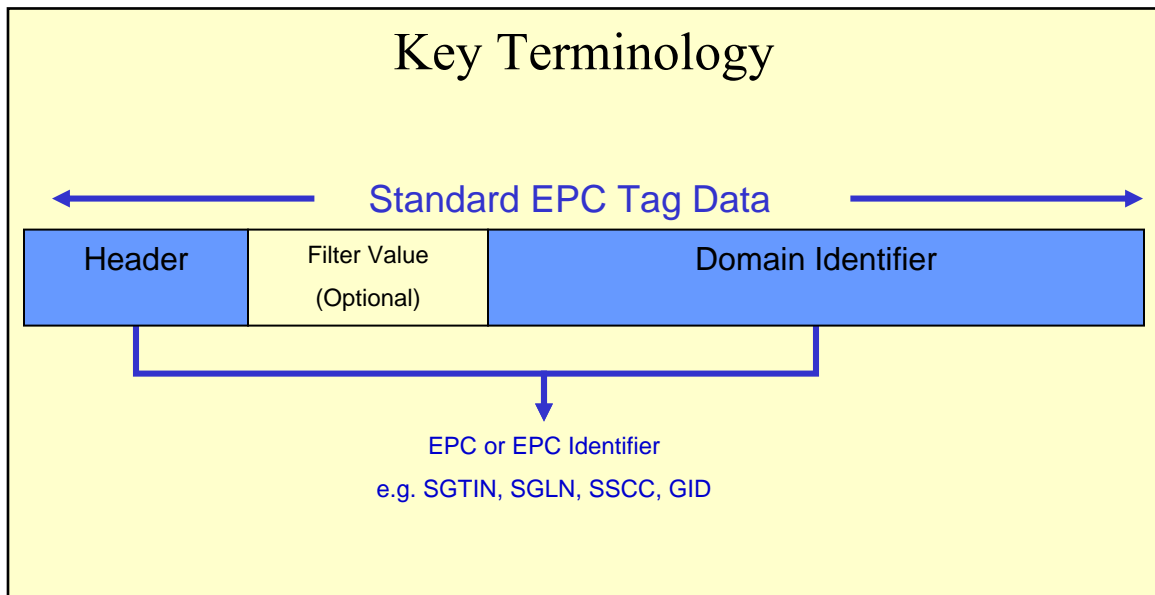


190 **1 Introduction**

191 The Electronic Product Code™ (EPC™) is an identification scheme for universally  
192 identifying physical objects via Radio Frequency Identification (RFID) tags and other  
193 means. The standardized EPC data consists of an EPC (or EPC Identifier) that uniquely  
194 identifies an individual object, as well as an optional Filter Value when judged to be  
195 necessary to enable effective and efficient reading of the EPC tags. In addition to this  
196 standardized data, certain Classes of EPC tags will allow user-defined data. The EPC  
197 Tag Data Standards will define the length and position of this data, without defining its  
198 content. Currently no user-defined data specifications exist since the related Class tags  
199 have not been defined.

200 The EPC Identifier is a meta-coding scheme designed to support the needs of various  
201 industries by accommodating both existing coding schemes where possible and defining  
202 new schemes where necessary. The various coding schemes are referred to as Domain  
203 Identifiers, to indicate that they provide object identification within certain domains such  
204 as a particular industry or group of industries. As such, the Electronic Product Code  
205 represents a family of coding schemes (or “namespaces”) and a means to make them  
206 unique across all possible EPC-compliant tags. These concepts are depicted in the chart  
207 below.

208



209

**Figure A.** EPC Terminology

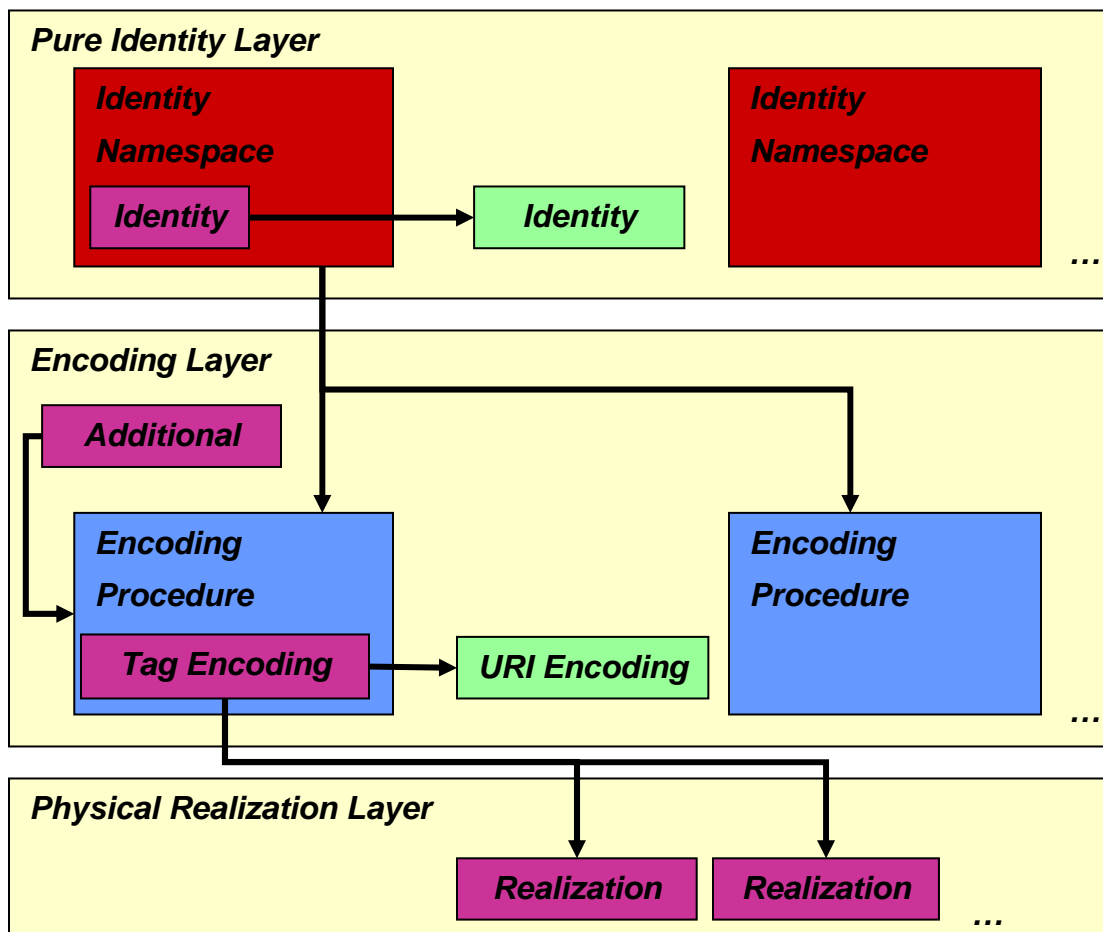
210

211 In this version of the EPC – EPC Version 1.1 – the specific coding schemes include a  
212 General Identifier (GID), a serialized version of the EAN.UCC Global Trade Item  
213 Number (GTIN®), the EAN.UCC Serial Shipping Container Code (SSCC®), the

214 EAN.UCC Global Location Number (GLN®), the EAN.UCC Global Returnable Asset  
 215 Identifier (GRAI®), and the EAN.UCC Global Individual Asset Identifier (GIAI®).  
 216 In the following sections, we will describe the structure and organization of the EPC and  
 217 provide illustrations to show its recommended use.  
 218 The EPCglobal Tag Data Standard V1.1 R1.27 has been approved by EAN.UCC with the  
 219 restrictions outlined in the General EAN.UCC Specifications Section 3.7, which is  
 220 excerpted into Tag Data Standard Appendix F.  
 221 The latest version of this specification can be [obtained](#) from EPCglobal.

## 222 2 Identity Concepts

223 To better understand the overall framework of the EPC Tag Data Standards, it's helpful  
 224 to distinguish between three levels of identification (See Figure B). Although this  
 225 specification addresses the pure identity and encoding layers in detail, all three layers are  
 226 described below to explain the layer concepts and the context for the encoding layer.



227

228 **Figure B.** Defined Identity Namespaces, Encodings, and Realizations.

- 229 • Pure identity -- the identity associated with a specific physical or logical entity,  
230 independent of any particular encoding vehicle such as an RF tag, bar code or  
231 database field. As such, a pure identity is an abstract name or number used to identify  
232 an entity. A pure identity consists of the information required to uniquely identify a  
233 specific entity, and no more. Identity URI – a representation of a pure identity as a  
234 Uniform Resource Identifier (URI). A URI is a character string representation that is  
235 commonly used to exchange identity data between software components of a larger  
236 system.
- 237 • Encoding -- a pure identity, together with additional information such as filter value,  
238 rendered into a specific syntax (typically consisting of value fields of specific sizes).  
239 A given pure identity may have a number of possible encodings, such as a Barcode  
240 Encoding, various Tag Encodings, and various URI Encodings. Encodings may also  
241 incorporate additional data besides the identity (such as the Filter Value used in some  
242 encodings), in which case the encoding scheme specifies what additional data it can  
243 hold.
- 244 • Physical Realization of an Encoding -- an encoding rendered in a concrete  
245 implementation suitable for a particular machine-readable form, such as a specific  
246 kind of RF tag or specific database field. A given encoding may have a number of  
247 possible physical realizations.

248 For example, the Serial Shipping Container Code (SSCC) format as defined by the  
249 EAN.UCC System is an example of a pure identity. An SSCC encoded into the EPC-  
250 SSCC 96-bit format is an example of an encoding. That 96-bit encoding, written onto a  
251 UHF Class 1 RF Tag, is an example of a physical realization.

252 A particular encoding scheme may implicitly impose constraints on the range of identities  
253 that may be represented using that encoding. For example, only 16,384 company  
254 prefixes can be encoded in the 64-bit SSCC scheme. In general, each encoding scheme  
255 specifies what constraints it imposes on the range of identities it can represent.

256 Conversely, a particular encoding scheme may accommodate values that are not valid  
257 with respect to the underlying pure identity type, thereby requiring an explicit constraint.  
258 For example, the EPC-SSCC 96-bit encoding provides 24 bits to encode a 7-digit  
259 company prefix. In a 24-bit field, it is possible to encode the decimal number 10,000,001,  
260 which is longer than 7 decimal digits. Therefore, this does not represent a valid SSCC,  
261 and is forbidden. In general, each encoding scheme specifies what limits it imposes on  
262 the value that may appear in any given encoded field.

## 263 **2.1 Pure Identities**

264 This section defines the pure identity types for which this document specifies encoding  
265 schemes.

### 266 **2.1.1 General Types**

267 This version of the EPC Tag Data Standards defines one general identity type. The  
268 *General Identifier (GID-96)* is independent of any known, existing specifications or

269 identity schemes. The General Identifier is composed of three fields - the *General*  
270 *Manager Number*, *Object Class* and *Serial Number*. Encodings of the GID include a  
271 fourth field, the header, to guarantee uniqueness in the EPC namespace.

272 The *General Manager Number* identifies an organizational entity (essentially a company,  
273 manager or other organization) that is responsible for maintaining the numbers in  
274 subsequent fields – Object Class and Serial Number. EPCglobal assigns the General  
275 Manager Number to an entity, and ensures that each General Manager Number is unique.

276 The *Object Class* is used by an EPC managing entity to identify a class or “type” of thing.  
277 These object class numbers, of course, must be unique within each General Manager  
278 Number domain. Examples of Object Classes could include case Stock Keeping Units of  
279 consumer-packaged goods or different structures in a highway system, like road signs,  
280 lighting poles, and bridges, where the managing entity is a County.

281 Finally, the *Serial Number* code, or serial number, is unique within each object class. In  
282 other words, the managing entity is responsible for assigning unique, non-repeating serial  
283 numbers for every instance within each object class.

## 284 **2.1.2 EAN.UCC System Identity Types**

285 This version of the EPC Tag Data Standards defines five EPC identity types derived from  
286 the EAN.UCC System family of product codes, each described in the subsections below.

287 EAN.UCC System codes have a common structure, consisting of a fixed number of  
288 decimal digits that encode the identity, plus one additional “check digit” which is  
289 computed algorithmically from the other digits. Within the non-check digits, there is an  
290 implicit division into two fields: a Company Prefix assigned by EAN or UCC to a  
291 managing entity, and the remaining digits, which are assigned by the managing entity.  
292 (The digits apart from the Company Prefix are called by a different name by each of the  
293 EAN.UCC System codes.) The number of decimal digits in the Company Prefix varies  
294 from 6 to 12 depending on the particular Company Prefix assigned. The number of  
295 remaining digits therefore varies inversely so that the total number of digits is fixed for a  
296 particular EAN.UCC System code type.

297 The EAN.UCC recommendations for the encoding of EAN.UCC System identities into  
298 bar codes, as well as for their use within associated data processing software, stipulate  
299 that the digits comprising a EAN.UCC System code should always be processed together  
300 as a unit, and not parsed into individual fields. This recommendation, however, is not  
301 appropriate within the EPC Network, as the ability to divide a code into the part assigned  
302 to the managing entity (the Company Prefix in EAN.UCC System types) versus the part  
303 that is managed by the managing entity (the remainder) is essential to the proper  
304 functioning of the Object Name Service (ONS). In addition, the ability to distinguish the  
305 Company Prefix is believed to be useful in filtering or otherwise securing access to EPC-  
306 derived data. Hence, the EPC encodings for EAN.UCC code types specified herein  
307 deviate from the aforementioned recommendations in the following ways:

- 308 • EPC encodings carry an explicit division between the Company Prefix and the  
309 remaining digits, with each individually encoded into binary. Hence, converting from

310 the traditional decimal representation of an EAN.UCC System code and an EPC  
 311 encoding requires independent knowledge of the length of the Company Prefix.

- 312 • EPC encodings do not include the check digit. Hence, converting from an EPC  
 313 encoding to a traditional decimal representation of a code requires that the check digit  
 314 be recalculated from the other digits.

315 **2.1.2.1 Serialized Global Trade Item Number (SGTIN)**

316 The Serialized Global Trade Item Number is a new identity type based on the EAN.UCC  
 317 Global Trade Item Number (GTIN) code defined in the General EAN.UCC  
 318 Specifications. A GTIN by itself does not fit the definition of an EPC pure identity,  
 319 because it does not uniquely identify a single physical object. Instead, a GTIN identifies  
 320 a particular class of object, such as a particular kind of product or SKU.

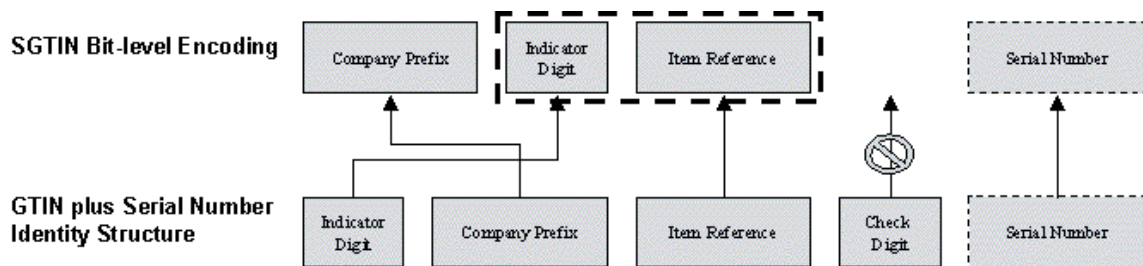
321 *All representations of SGTIN support the full 14-digit GTIN format. This means that the*  
 322 *zero indicator-digit and leading zero in the Company Prefix for UCC-12, and the zero*  
 323 *indicator-digit for EAN/UCC-13, can be encoded and interpreted accurately from an*  
 324 *EPC encoding. EAN/UCC-8 is not currently supported in EPC, but would be supported*  
 325 *in full 14-digit GTIN format as well.*

326 To create a unique identifier for individual objects, the GTIN is augmented with a serial  
 327 number, which the managing entity is responsible for assigning uniquely to individual  
 328 object classes. The combination of GTIN and a unique serial number is called a  
 329 Serialized GTIN (SGTIN).

330 The SGTIN consists of the following information elements:

- 331 • The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company  
 332 Prefix is the same as the Company Prefix digits within an EAN.UCC GTIN decimal  
 333 code.
- 334 • The *Item Reference*, assigned by the managing entity to a particular object class. The  
 335 Item Reference for the purposes of EPC encoding is derived from the GTIN by  
 336 concatenating the Indicator Digit of the GTIN and the Item Reference digits, and  
 337 treating the result as a single integer.
- 338 • The *Serial Number*, assigned by the managing entity to an individual object. The  
 339 serial number is not part of the GTIN code, but is formally a part of the SGTIN.

340



341  
 342

343 **Figure C.** How the parts of the decimal SGTIN are extracted, rearranged, and augmented for  
344 encoding.

345 The SGTIN is not explicitly defined in the EAN.UCC General Specifications. However,  
346 it may be considered equivalent to a UCC/EAN-128 bar code that contains both a GTIN  
347 (Application Identifier 01) and a serial number (Application Identifier 21). Serial  
348 numbers in AI 21 consist of one to twenty characters, where each character can be a digit,  
349 uppercase or lowercase letter, or one of a number of allowed punctuation characters. The  
350 complete AI 21 syntax is supported by the pure identity URI syntax specified in  
351 Section 4.3.3.

352 When representing serial numbers in 64- and 96-bit tags, however, only a subset of the  
353 serial number allowed in the General EAN.UCC Specifications for Application Identifier  
354 21 are permitted. Specifically, the permitted serial numbers are those consisting of one or  
355 more digits characters, with no leading zeros, and whose value when considered as an  
356 integer fits within the range restrictions of the 64- and 96-bit tag encodings.

357 While these limitations exist for 64- and 96-bit tag encodings, future tag encodings may  
358 allow a wider range of serial numbers. Therefore, application authors and database  
359 designers should take the EAN.UCC specifications for Application Identifier 21 into  
360 account in order to accommodate further expansions of the Tag Data Standard.

361 *Explanation (non-normative): The restrictions are necessary for 64- and 96-bit tags in*  
362 *order for serial numbers to fit within the small number of bits we have available. So we*  
363 *restrict the range, and also disallow alphabetic characters. The reason we also forbid*  
364 *leading zeros is that on these tags we're encoding the serial number value by considering*  
365 *it to be a decimal integer then encoding the integer value in binary. By considering it to*  
366 *be a decimal integer, we can't distinguish between "00034", "034", or "34" (for example)*  
367 *-- they all have the same value when considered as an integer rather than a character*  
368 *string. In order to insure that every encoded value can be decoded uniquely, we*  
369 *arbitrarily say that serial numbers can't have leading zeros. Then, when we see the bits*  
370 *00000000000000000000000010010 on the tag, we decode the serial number as "34" (not*  
371 *"034" or "00034").*

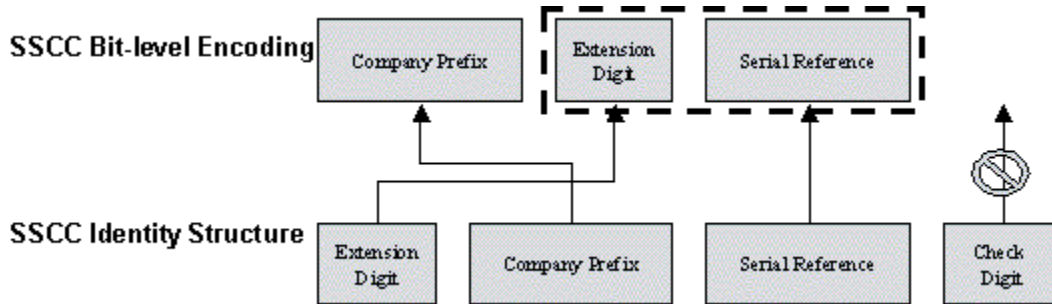
### 372 **2.1.2.2 Serial Shipping Container Code (SSCC)**

373 The Serial Shipping Container Code (SSCC) is defined by the General EAN.UCC  
374 Specifications. Unlike the GTIN, the SSCC is already intended for assignment to  
375 individual objects and therefore does not require any additional fields to serve as an EPC  
376 pure identity.

377 *Note that many applications of SSCC have historically included the Application Identifier*  
378 *(00) in the SSCC identifier field when stored in a database. This is not a standard*  
379 *requirement, but a widespread practice. The Application Identifier is a sort of header*  
380 *used in bar code applications, and can be inferred directly from EPC headers*  
381 *representing SSCC. In other words, an SSCC EPC can be interpreted as needed to*  
382 *include the (00) as part of the SSCC identifier or not.*

383 The SSCC consists of the following information elements:

- 384 • The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company  
385 Prefix is the same as the Company Prefix digits within an EAN.UCC SSCC decimal  
386 code.
  - 387 • The *Serial Reference*, assigned uniquely by the managing entity to a specific shipping  
388 unit. The Serial Reference for the purposes of EPC encoding is derived from the  
389 SSCC by concatenating the Extension Digit of the SSCC and the Serial Reference  
390 digits, and treating the result as a single integer.
- 391



392  
393 **Figure D.** How the parts of the decimal SSCC are extracted and rearranged for encoding.

### 394 2.1.2.3 Serialized Global Location Number (SGLN)

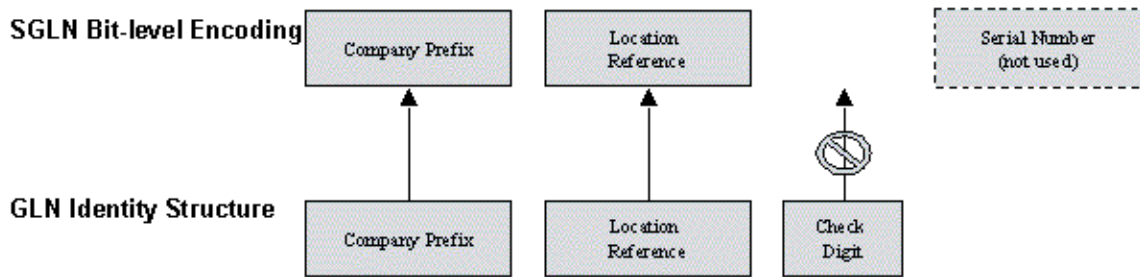
395 The Global Location Number (GLN) is defined by the General EAN.UCC Specifications.  
396 A GLN can represent either a discrete, unique physical location such as a dock door or a  
397 warehouse slot, or an aggregate physical location such as an entire warehouse. In  
398 addition, a GLN can represent a logical entity such as an “organization” that performs a  
399 business function such as placing an order.

400 Recognizing these variables, the EPC GLN is meant to apply only to the physical  
401 location sub-type of GLN.

- 402 ➤ The serial number field is reserved and should not be used, until the EAN.UCC  
403 community determines the appropriate way, if any, for extending GLN.

404 The SGLN consists of the following information elements:

- 405 • The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company  
406 Prefix is the same as the Company Prefix digits within an EAN.UCC GLN decimal  
407 code.
- 408 • The *Location Reference*, assigned uniquely by the managing entity to an aggregate or  
409 specific physical location.
- 410 • The *Serial Number*, assigned by the managing entity to an individual unique location.  
411 ➤ The serial number should not be used until specified by the EAN.UCC General  
412 Specifications .



413

414 **Figure E.** How the parts of the decimal SGLN are extracted and rearranged for encoding.

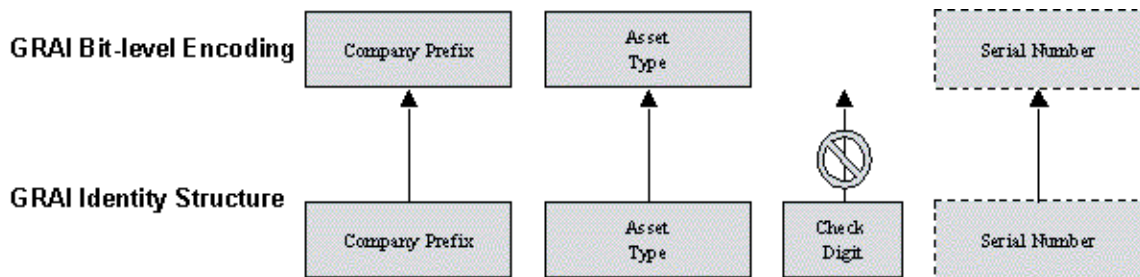
415 **2.1.2.4 Global Returnable Asset Identifier (GRAI)**

416 The Global Returnable Asset Identifier (GRAI) is defined by the General EAN.UCC  
 417 Specifications. Unlike the GTIN, the GRAI is already intended for assignment to  
 418 individual objects and therefore does not require any additional fields to serve as an EPC  
 419 pure identity.

420

421 The GRAI consists of the following information elements:

- 422 • The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company  
 423 Prefix is the same as the Company Prefix digits within an EAN.UCC GRAI decimal  
 424 code.
- 425 • The *Asset Type*, assigned by the managing entity to a particular class of asset.
- 426 • The *Serial Number*, assigned by the managing entity to an individual object. The EPC  
 427 representation is only capable of representing a subset of Serial Numbers allowed in  
 428 the General EAN.UCC Specifications. Specifically, only those Serial Numbers  
 429 consisting of one or more digits, with no leading zeros, are permitted [see Appendix F  
 430 for details].



431

432 **Figure F.** How the parts of the decimal GRAI are extracted and rearranged for encoding.

433 **2.1.2.5 Global Individual Asset Identifier (GIAI)**

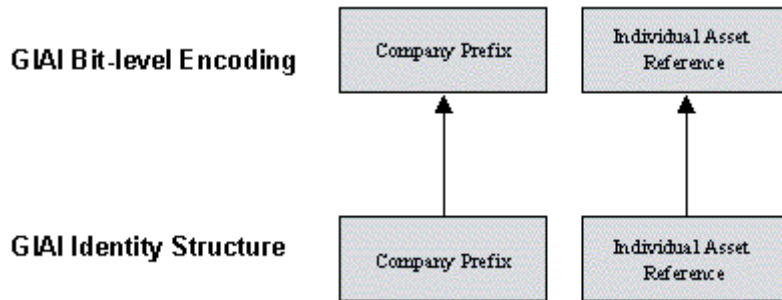
434 The Global Individual Asset Identifier (GIAI) is defined by the General EAN.UCC  
 435 Specifications. Unlike the GTIN, the GIAI is already intended for assignment to  
 436 individual objects and therefore does not require any additional fields to serve as an EPC  
 437 pure identity.

438



439 The GIAI consists of the following information elements:

- 440 • The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company  
441 Prefix is the same as the Company Prefix digits within an EAN.UCC GIAI decimal  
442 code.
- 443 • The *Individual Asset Reference*, assigned uniquely by the managing entity to a  
444 specific asset. The EPC representation is only capable of representing a subset of  
445 Individual Asset References allowed in the General EAN.UCC Specifications.  
446 Specifically, only those Individual Asset References consisting of one or more digits,  
447 with no leading zeros, are permitted.



448

449 **Figure G.** How the parts of the decimal GIAI are extracted and rearranged for encoding.

### 450 **2.1.3 DoD Identity Type**

451 The DoD Construct identifier is defined by the United States Department of Defense.

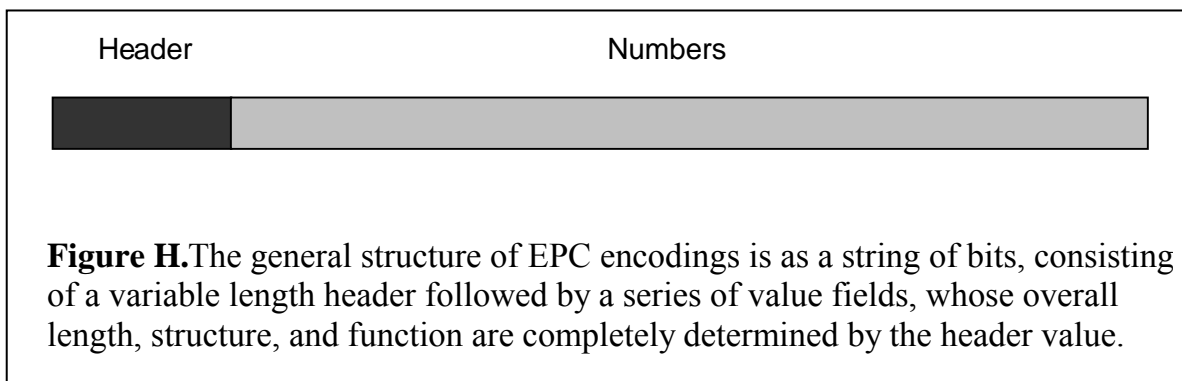
452 This tag data construct may be used to encode 64-bit and 96-bit Class 0 and Class 1 tags  
453 for shipping goods to the United States Department of Defense by a supplier who has  
454 already been assigned a CAGE (Commercial and Government Entity) code.

455 At the time of this writing, the details of what information to encode into these fields is  
456 explained in a document titled "United States Department of Defense Supplier's Passive  
457 RFID Information Guide" that can be obtained at the United States Department of  
458 Defense's web site (<http://www.dodrfid.org/supplierguide.htm>).

## 459 **3 EPC Tag Bit-level Encodings**

460 The general structure of EPC encodings on a tag is as a string of bits (i.e., a binary  
461 representation), consisting of a tiered, variable length header followed by a series of  
462 numeric fields (Figure H) whose overall length, structure, and function are completely  
463 determined by the header value.

464



**Figure H.** The general structure of EPC encodings is as a string of bits, consisting of a variable length header followed by a series of value fields, whose overall length, structure, and function are completely determined by the header value.

### 465 3.1 Headers

466 As previously stated, the Header defines the overall length, identity type, and structure of  
 467 the EPC Tag Encoding, including its Filter Value, if any. The header is of variable length,  
 468 using a tiered approach in which a zero value in each tier indicates that the header is  
 469 drawn from the next longer tier. For the encodings defined in this specification, headers  
 470 are either 2 bits or 8 bits. Given that a zero value is reserved to indicate a header in the  
 471 next longer tier, the 2-bit header can have 3 possible values (01, 10, and 11, not 00), and  
 472 the 8-bit header can have 63 possible values (recognizing that the first 2 bits must be 00  
 473 and 00000000 is reserved to allow headers that are longer than 8 bits).

474 *Explanation (non-normative): The tiered scheme is designed to simplify the Header*  
 475 *processing required by the Reader in order to determine the tag data format, particularly*  
 476 *the location of the Filter Value, while attempting to conserve bits for data values in the*  
 477 *64-bit tag. In the not-too-distant future, we expect to be able to “reclaim” the 2-bit tier*  
 478 *when 64-bit tags are no longer needed, thereby expanding the 8-bit Header from 63*  
 479 *possible values to 255.*

480 The assignment of Header values has been designed so that the tag length may be easily  
 481 discerned by examining the leftmost (or Preamble) bits of the Header. Moreover, the  
 482 design is aimed at having as few Preambles per tag length as possible, ideally 1 but  
 483 certainly no more than 2 or 3. This latter objective prompts us to avoid, if it all possible,  
 484 using those Preambles that allow very few Header values (as noted in italics in Table 1  
 485 below). The purpose of this Preamble-to-Tag-Length design is so that RFID readers may  
 486 easily determine a tag’s length. See Appendix B for a detailed discussion of why this is  
 487 important.

488 The currently assigned Headers are such that a tag may be inferred to be 64 bits if either  
 489 the first two bits are non-zero or the first five bits are equal to 00001; otherwise, the  
 490 Header indicates the tag is 96 bits. In the future, unassigned Headers may be assigned for  
 491 these and other tag lengths.

492 Certain Preambles aren’t currently tied to a particular tag length to leave open the option  
 493 for additional tag lengths, especially longer ones that can accommodate longer coding  
 494 schemes such as the Unique ID (UID) being pursued by suppliers to the US Department  
 495 of Defense.

496 Thirteen encoding schemes have been defined in this version of the EPC Tag Data  
 497 Standard, as shown in Table 1 below.

<b>Header Value (binary)</b>	<b>Tag Length (bits)</b>	<b>Encoding Scheme</b>
01	64	[Reserved 64-bit scheme]
10	64	SGTIN-64
1100 0000 ... 1100 1101	64	[Reserved 64-bit scheme]
1100 1110	64	DoD-64
1100 1111 ... 1111 1111	64	[Reserved 64-bit scheme]
0000 0001	<i>na</i>	<i>[1 reserved scheme]</i>
0000 001x	<i>na</i>	<i>[2 reserved schemes]</i>
0000 01xx	<i>na</i>	<i>[4 reserved schemes]</i>
0000 1000	64	SSCC-64
0000 1001	64	GLN-64
0000 1010	64	GRAI-64
0000 1011	64	GIAI-64
0000 1100 ... 0000 1111	64	[4 reserved 64-bit schemes]
0001 0000 ... 0010 1110	na	[31 reserved schemes]
0010 1111	96	DoD-96
0011 0000	96	SGTIN-96
0011 0001	96	SSCC-96
0011 0010	96	GLN-96
0011 0011	96	GRAI-96
0011 0100	96	GIAI-96

Header Value (binary)	Tag Length (bits)	Encoding Scheme
0011 0101	96	GID-96
0011 0110	96	[10 reserved 96-bit schemes]
...		
0011 1111		
0000 0000 ...		[reserved for future headers longer than 8 bits]

**Table 1.** Electronic Product Code Headers

498

499

### 3.2 Notational Conventions

500

501 In the remainder of this section, tag-encoding schemes are depicted using the following  
502 notation (See Table 2).

	Header	Filter Value	Company Prefix <i>Index</i>	Item Reference	Serial Number
SGTIN-64	2	3	14	20	25
	10 (Binary value)	(Refer to Table 5 for values)	16,383 (Max. decimal value)	9 -1,048,575 (Max. decimal range*)	33,554,431 (Max. decimal value)

\*Max. decimal value range of Item Reference field varies with the length of the Company Prefix

503

**Table 2.** Example of Notation Conventions.

504

505

506 The first column of the table gives the formal name for the encoding. The remaining  
507 columns specify the layout of each field within the encoding. The field in the leftmost  
508 column occupies the most significant bits of the encoding (this is always the header field),  
509 and the field in the rightmost column occupies the least significant bits. Each field is a  
510 non-negative integer, encoded into binary using a specified number of bits. Any unused  
511 bits (i.e., bits not required by a defined field) are explicitly indicated in the table, so that  
512 the columns in the table are concatenated with no gaps to form the complete binary  
513 encoding.

514 Reading down each column, the table gives the formal name of the field, the number of  
515 bits used to encode the field's value, and the value or range of values for the field. The  
516 value may represent one of the following:

- 517 • The value of a binary number indicated by (*Binary value*), as is the case for the  
518 Header field in the example table above
- 519 • The maximum decimal value indicated by (*Max. decimal value*) of a fixed length  
520 field. This is calculated as  $2^n - 1$ , where  $n$  = the fixed number of bits in the field.
- 521 • A range of maximum decimal values indicated by (*Max. decimal range*). This  
522 range is calculated using the normative rules expressed in the related encoding  
523 procedure section
- 524 • A reference to a table that provides the valid values defined for the field..

525 In some cases, the number of possible values in one field depends on the specific value  
526 assigned to another field. In such cases, a range of maximum decimal values is shown. In  
527 the example above, the maximum decimal value for the Item Reference field depends on  
528 the length of the Company Prefix field; hence the maximum decimal value is shown as a  
529 range. Where a field must contain a specific value (as in the Header field), the last row of  
530 the table specifies the specific value rather than the number of possible values.

531 Some encodings have fields that are of variable length. The accompanying text specifies  
532 how the field boundaries are determined in those cases.

533 Following an overview of each encoding scheme are a detailed encoding procedure and  
534 decoding procedure. The encoding and decoding procedure provide the normative  
535 specification for how each type of encoding is to be formed and interpreted.

### 536 3.3 General Identifier (GID-96)

537 The *General Identifier* is defined for a 96-bit EPC, and is independent of any existing  
538 identity specification or convention. The General Identifier is composed of three fields -  
539 the *General Manager Number*, *Object Class* and *Serial Number*. Encodings of the GID  
540 include a fourth field, the header, to guarantee uniqueness in the EPC namespace, as  
541 shown in Table 3.

542

	Header	General Manager Number	Object Class	Serial Number
GID-96	8	28	24	36
	0011 0101 (Binary value)	268,435,455 (Max. decimal value)	16,777,215 (Max. decimal value)	68,719,476,735 (Max. decimal value)

543 **Table 3.** The General Identifier (GID-96) includes three fields in addition to the header – the  
544 *General Manager Number*, *Object class* and *Serial Number* numbers.

545

546 The *General Manager Number* identifies essentially a company, manager or  
547 organization; that is an entity responsible for maintaining the numbers in subsequent  
548 fields – Object Class and Serial Number. EPCglobal assigns the General Manager  
549 Number to an entity, and ensures that each General Manager Number is unique.

550 The third component is *Object Class*, and is used by an EPC managing entity to identify a  
551 class or “type” of thing. These object class numbers, of course, must be unique within  
552 each General Manager Number domain. Examples of Object Classes could include case  
553 Stock Keeping Units of consumer-packaged goods and component parts in an assembly.

554 Finally, the *Serial Number* code, or serial number, is unique within each object class. In  
555 other words, the managing entity is responsible for assigning unique – non-repeating  
556 serial numbers for every instance within each object class code.

### 557 **3.3.1.1 GID-96 Encoding Procedure**

558 The following procedure creates a GID-96 encoding.

559 Given:

560 An General Manager Number  $M$  where  $0 \leq M < 2^{28}$

561 An Object Class  $C$  where  $0 \leq C < 2^{24}$

562 A Serial Number  $S$  where  $0 \leq S < 2^{36}$

563 Procedure:

564 1. Construct the final encoding by concatenating the following bit fields, from most  
565 significant to least significant: Header 00110101, General Manager Number  $M$  (28 bits),  
566 Object Class  $C$  (24 bits), Serial Number  $S$  (36 bits).

### 567 **3.3.1.2 GID-96 Decoding Procedure**

568 Given:

569 A GID-96 as a 96-bit string  $00110101b_{87}b_{86}\dots b_0$  (where the first eight bits 00110101 are  
570 the header)

571 Yields:

572 An General Manager Number

573 An Object Class

574 A Serial Number

575 Procedure:

576 1. Bits  $b_{87}b_{86}\dots b_{60}$ , considered as an unsigned integer, are the General Manager Number.

577 2. Bits  $b_{59}b_{58}\dots b_{36}$ , considered as an unsigned integer, are the Object Class.

578 3. Bits  $b_{35}b_{34}\dots b_0$ , considered as an unsigned integer, are the Serial Number.

579 **3.4 Serialized Global Trade Item Number (SGTIN)**

580 The EPC encoding scheme for SGTIN permits the direct embedding of EAN.UCC  
 581 System standard GTIN and Serial Number codes on EPC tags. In all cases, the check  
 582 digit is not encoded. Two encoding schemes are specified, SGTIN-64 (64 bits) and  
 583 SGTIN-96 (96 bits).

584 In the SGTIN-64 encoding, the limited number of bits prohibits a literal embedding of the  
 585 GTIN. As a partial solution, a Company Prefix *Index* is used. This Index, which can  
 586 accommodate up to 16,384 codes, is assigned to companies that need to use the 64 bit  
 587 tags, in addition to their existing EAN.UCC Company Prefixes. The Index is encoded on  
 588 the tag instead of the Company Prefix, and is subsequently translated to the Company  
 589 Prefix at low levels of the EPC system components (i.e. the Reader or Savant). While  
 590 this means that only a limited number of Company Prefixes can be represented in the 64-  
 591 bit tag, this is a transitional step to full accommodation in 96-bit and additional encoding  
 592 schemes. The 64-bit company prefix index table can be found at <http://www.onsepc.com>.

593 **3.4.1 SGTIN-64**

594 The SGTIN-64 includes *five* fields – *Header*, *Filter Value*, *Company Prefix Index*, *Item*  
 595 *Reference*, and *Serial Number*, as shown in Table 4.

596

	Header	Filter Value	Company Prefix Index	Item Reference	Serial Number
SGTIN-64	2	3	14	20	25
	10 (Binary value)	(Refer to Table 5 for values)	16,383 (Max. decimal value)	9 -1,048,575 (Max. decimal range*)	33,554,431 (Max. decimal value)

597 \*Max. decimal value range of Item Reference field varies with the length of the Company Prefix

598 **Table 4.** The EPC SGTIN-64 bit allocation, header, and maximum decimal values.

- 599
- *Header* is 2 bits, with a binary value of 10.
  - *Filter Value* is not part of the SGTIN pure identity, but is additional data that is used for fast filtering and pre-selection of basic logistics types. The Filter Values for 64-bit and 96-bit SGTIN are the same. The normative specifications for Filter Values are specified in Table 5. The value of 000 means “All Others”. That is, a filter value of 000 means that the object to which the tag is affixed does not match any of the logistic types defined as other filter values in this specification. It should be noted that tags conforming to earlier versions of this specification, in which 000 was the only value approved for use, will have filter value equal to 000 regardless of the logistic types, but following the ratification of this standard, the filter value should be set to match the object to which the tag is affixed, and use 000 only if the filter value for
- 600  
601  
602  
603  
604  
605  
606  
607  
608  
609

610 such object does not exist in the specification. A Standard Trade Item grouping  
 611 represents all levels of packaging for logistical units. The Single Shipping /  
 612 Consumer Trade item type should be used when the individual item is also the  
 613 logistical unit (e.g. Large screen television, Bicycle).

614

Type	Binary Value
All Others	000
Retail Consumer Trade Item	001
Standard Trade Item Grouping	010
Single Shipping/ Consumer Trade Item	011
Reserved	100
Reserved	101
Reserved	110
Reserved	111

615 **Table 5.** SGTIN Filter Values .

616

- 617 • *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this  
 618 field is not the Company Prefix itself, but rather an index into a table that provides the  
 619 Company Prefix as well as an indication of the Company Prefix’s length. The means  
 620 by which hardware or software may obtain the contents of the translation table is  
 621 specified in [Translation of 64-bit Tag Encoding Company Prefix Indices Into  
 622 EAN.UCC Company Prefixes].
- 623 • *Item Reference* encodes the GTIN Item Reference number and Indicator Digit. The  
 624 Indicator Digit is combined with the Item Reference field in the following manner:  
 625 Leading zeros on the item reference are significant. Put the Indicator Digit in the  
 626 leftmost position available within the field. *For instance, 00235 is different than 235.*  
 627 *With the indicator digit of 1, the combination with 00235 is 100235.* The resulting  
 628 combination is treated as a single integer, and encoded into binary to form the Item  
 629 Reference field.
- 630 • *Serial Number* contains a serial number. The SGTIN-64 encoding is only capable of  
 631 representing integer-valued serial numbers with limited range. Other EAN.UCC  
 632 specifications permit a broader range of serial numbers. In particular, the EAN-128  
 633 barcode symbology provides for a 20-character alphanumeric serial number to be  
 634 associated with a GTIN using Application Identifier (AI) 21 [EANUCCGS]. It is  
 635 possible to convert between the serial numbers in the SGTIN-64 tag encoding and the  
 636 serial numbers in AI 21 barcodes under certain conditions. Specifically, such  
 637 interconversion is possible when the alphanumeric serial number in AI 21 happens to  
 638 consist only of digit characters, with no leading zeros, and whose value when



639 interpreted as an integer falls within the range limitations of the SGTIN-64 tag  
640 encoding. These considerations are reflected in the encoding and decoding  
641 procedures below.

#### 642 **3.4.1.1 SGTIN-64 Encoding Procedure**

643 The following procedure creates an SGTIN-64 encoding.

644 Given:

- 645 • An EAN.UCC GTIN-14 consisting of digits  $d_1d_2\dots d_{14}$
- 646 • The length  $L$  of the company prefix portion of the GTIN
- 647 • A Serial Number  $S$  where  $0 \leq S < 2^{25}$ , or an UCC/EAN-128 Application Identifier 21  
648 consisting of characters  $s_1s_2\dots s_K$ .
- 649 • A Filter Value  $F$  where  $0 \leq F < 8$

650 Procedure:

- 651 1. Extract the EAN.UCC Company Prefix  $d_2d_3\dots d_{(L+1)}$
- 652 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table  
653 to obtain the corresponding Company Prefix Index,  $C$ . If the Company Prefix was not  
654 found in the Company Prefix Translation Table, stop: this GTIN cannot be encoded in the  
655 SGTIN-64 encoding.
- 656 3. Construct the Item Reference + Indicator Digit by concatenating digits  
657  $d_1d_{(L+2)}d_{(L+3)}\dots d_{13}$  and considering the result to be a decimal integer,  $I$ . If  $I \geq 2^{20}$ , stop:  
658 this GTIN cannot be encoded in the SGTIN-64 encoding.
- 659 4. When the Serial Number is provided directly as an integer  $S$  where  $0 \leq S < 2^{25}$ ,  
660 proceed to Step 5. Otherwise, when the Serial Number is provided as an UCC/EAN-128  
661 Application Identifier 21 consisting of characters  $s_1s_2\dots s_K$ , construct the Serial Number  
662 by concatenating digits  $s_1s_2\dots s_K$ . If any of these characters is not a digit, stop: this Serial  
663 Number cannot be encoded in the SGTIN-64 encoding. Also, if  $K > 1$  and  $s_1 = 0$ , stop:  
664 this Serial Number cannot be encoded in the SGTIN-64 encoding (because leading zeros  
665 are not permitted except in the case where the Serial Number consists of a single zero  
666 digit). Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^{25}$ , stop: this  
667 Serial Number cannot be encoded in the SGTIN-64 encoding.
- 668 5. Construct the final encoding by concatenating the following bit fields, from most  
669 significant to least significant: Header 10 (2 bits), Filter Value  $F$  (3 bits), Company  
670 Prefix Index  $C$  from Step 2 (14 bits), Item Reference from Step 3 (20 bits), Serial  
671 Number  $S$  from Step 4 (25 bits).

#### 672 **3.4.1.2 SGTIN-64 Decoding Procedure**

673 Given:

- 674 • An SGTIN-64 as a 64-bit bit string  $10b_{61}b_{60}\dots b_0$  (where the first two bits 10 are the  
675 header)

676 Yields:

- 677 • An EAN.UCC GTIN-14
- 678 • A Serial Number
- 679 • A Filter Value

680 Procedure:

- 681 1. Bits  $b_{61}b_{60}b_{59}$ , considered as an unsigned integer, are the Filter Value.
- 682 2. Extract the Company Prefix Index  $C$  by considering bits  $b_{58}b_{57}\dots b_{45}$  as an unsigned  
683 integer.
- 684 3. Look up the Company Prefix Index  $C$  in the Company Prefix Translation Table to  
685 obtain the EAN.UCC Company Prefix  $p_1p_2\dots p_L$  consisting of  $L$  decimal digits (the value  
686 of  $L$  is also obtained from the table). If the Company Prefix Index  $C$  is not found in the  
687 Company Prefix Translation Table, stop: this bit string cannot be decoded as an SGTIN-  
688 64.
- 689 4. Consider bits  $b_{44}b_{43}\dots b_{25}$  as an unsigned integer. If this integer is greater than or  
690 equal to  $10^{(13-L)}$ , stop: the input bit string is not a legal SGTIN-64 encoding. Otherwise,  
691 convert this integer to a (13-L)-digit decimal number  $i_1i_2\dots i_{(13-L)}$ , adding leading zeros as  
692 necessary to make (13-L) digits.
- 693 5. Construct a 13-digit number  $d_1d_2\dots d_{13}$  where  $d_1 = i_1$  from Step 4,  $d_2d_3\dots d_{(L+1)} =$   
694  $p_1p_2\dots p_L$  from Step 3, and  $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2i_3\dots i_{(13-L)}$  from Step 4.
- 695 6. Calculate the check digit  $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 +$   
696  $d_8 + d_{10} + d_{12})) \bmod 10$ .
- 697 7. The EAN.UCC GTIN-14 is the concatenation of digits from Steps 5 and 6:  $d_1d_2\dots d_{14}$ .
- 698 8. Bits  $b_{24}b_{23}\dots b_0$ , considered as an unsigned integer, are the Serial Number.
- 699 9. (Optional) If it is desired to represent the serial number as a UCC/EAN-128  
700 Application Identifier 21, convert the integer from Step 8 to a decimal string with no  
701 leading zeros. If the integer in Step 8 is zero, convert it to a string consisting of the single  
702 character "0".

### 703 **3.4.2 SGTIN-96**

704 In addition to a Header, the SGTIN-96 is composed of five fields: the *Filter Value*,  
705 *Partition*, *Company Prefix*, *Item Reference*, and *Serial Number*, as shown in Table 6.

	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
SGTIN-96	8	3	3	20-40	24-4	38
	0011 0000 (Binary value)	(Refer to Table 5 for values)	(Refer to Table 7 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	9,999,999 – 9 (Max. decimal range*)	274,877,906,943 (Max. decimal value)

706 \*Max. decimal value range of Company Prefix and Item Reference fields vary according to the contents of  
707 the Partition field.

708 **Table 6.** The EPC SGTIN-96 bit allocation, header, and maximum decimal values.

- 709 • *Header* is 8-bits, with a binary value of 0011 0000.
- 710 • *Filter Value* is not part of the GTIN or EPC identifier, but is used for fast filtering and  
711 pre-selection of basic logistics types. The Filter Values for 64-bit and 96-bit GTIN  
712 are the same. See Table 5.
- 713 • *Partition* is an indication of where the subsequent Company Prefix and Item  
714 Reference numbers are divided. This organization matches the structure in the  
715 EAN.UCC GTIN in which the Company Prefix added to the Item Reference number  
716 (plus the single Indicator Digit) totals 13 digits, yet the Company Prefix may vary  
717 from 6 to 12 digits and the Item Reference (including the single Indicator Digit) from  
718 7 to 1 digit(s). The available values of *Partition* and the corresponding sizes of the  
719 *Company Prefix* and *Item Reference* fields are defined in Table 7.
- 720 • *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.
- 721 • *Item Reference* contains a literal embedding of the GTIN Item Reference number.  
722 The Indicator Digit is combined with the Item Reference field in the following  
723 manner: Leading zeros on the item reference are significant. Put the Indicator Digit in  
724 the leftmost position available within the field. *For instance, 00235 is different than*  
725 *235. With the indicator digit of 1, the combination with 00235 is 100235.* The  
726 resulting combination is treated as a single integer, and encoded into binary to form  
727 the *Item Reference* field.
- 728 • *Serial Number* contains a serial number. The SGTIN-96 encoding is only capable of  
729 representing integer-valued serial numbers with limited range. Other EAN.UCC  
730 specifications permit a broader range of serial numbers. In particular, the EAN-128  
731 barcode symbology provides for a 20-character alphanumeric serial number to be  
732 associated with a GTIN using Application Identifier (AI) 21 [EANUCCGS]. It is  
733 possible to convert between the serial numbers in the SGTIN-96 tag encoding and the  
734 serial numbers in AI 21 barcodes under certain conditions. Specifically, such  
735 interconversion is possible when the alphanumeric serial number in AI 21 happens to

736 consist only of digit characters, with no leading zeros, and whose value when  
 737 interpreted as an integer falls within the range limitations of the SGTIN-96 tag  
 738 encoding. These considerations are reflected in the encoding and decoding  
 739 procedures below.  
 740

Partition Value ( <i>P</i> )	Company Prefix		Item Reference and Indicator Digit	
	Bits ( <i>M</i> )	Digits ( <i>L</i> )	Bits ( <i>N</i> )	Digits
0	40	12	4	1
1	37	11	7	2
2	34	10	10	3
3	30	9	14	4
4	27	8	17	5
5	24	7	20	6
6	20	6	24	7

741 • **Table 7.** SGTIN-96 Partitions.

742 **3.4.2.1 SGTIN-96 Encoding Procedure**

743 The following procedure creates an SGTIN-96 encoding.

744 Given:

- 745 • An EAN.UCC GTIN-14 consisting of digits  $d_1d_2\dots d_{14}$
- 746 • The length  $L$  of the Company Prefix portion of the GTIN
- 747 • A Serial Number  $S$  where  $0 \leq S < 2^{38}$ , or an UCC/EAN-128 Application Identifier 21
- 748 consisting of characters  $s_1s_2\dots s_K$ .
- 749 • A Filter Value  $F$  where  $0 \leq F < 8$

750 Procedure:

- 751 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column
- 752 of the Partition Table (Table 7) to determine the Partition Value,  $P$ , the number of bits  $M$
- 753 in the Company Prefix field, and the number of bits  $N$  in the Item Reference and
- 754 Indicator Digit field. If  $L$  is not found in any row of Table 7, stop: this GTIN cannot be
- 755 encoded in an SGTIN-96.
- 756 2. Construct the Company Prefix by concatenating digits  $d_2d_3\dots d_{(L+1)}$  and considering
- 757 the result to be a decimal integer,  $C$ .

758 3. Construct the Item Reference + Indicator Digit by concatenating digits  
759  $d_1d_{(L+2)}d_{(L+3)}\dots d_{13}$  and considering the result to be a decimal integer,  $I$ .

760 4. When the Serial Number is provided directly as an integer  $S$  where  $0 \leq S < 2^{38}$ ,  
761 proceed to Step 5. Otherwise, when the Serial Number is provided as an UCC/EAN-128  
762 Application Identifier 21 consisting of characters  $s_1s_2\dots s_K$ , construct the Serial Number  
763 by concatenating digits  $s_1s_2\dots s_K$ . If any of these characters is not a digit, stop: this Serial  
764 Number cannot be encoded in the SGTIN-96 encoding. Also, if  $K > 1$  and  $s_1 = 0$ , stop:  
765 this Serial Number cannot be encoded in the SGTIN-96 encoding (because leading zeros  
766 are not permitted except in the case where the Serial Number consists of a single zero  
767 digit). Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^{38}$ , stop: this  
768 Serial Number cannot be encoded in the SGTIN-96 encoding.

769 5. Construct the final encoding by concatenating the following bit fields, from most  
770 significant to least significant: Header 00110000 (8 bits), Filter Value  $F$  (3 bits),  
771 Partition Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Item  
772 Reference from Step 3 ( $N$  bits), Serial Number  $S$  from Step 4 (38 bits). Note that  $M+N =$   
773 44 bits for all  $P$ .

#### 774 3.4.2.2 SGTIN-96 Decoding Procedure

775 Given:

- 776 • An SGTIN-96 as a 96-bit bit string  $00110000b_{87}b_{86}\dots b_0$  (where the first eight bits  
777  $00110000$  are the header)

778 Yields:

- 779 • An EAN.UCC GTIN-14
- 780 • A Serial Number
- 781 • A Filter Value

782 Procedure:

- 783 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 784 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
785  $P = 7$ , stop: this bit string cannot be decoded as an SGTIN-96.
- 786 3. Look up the Partition Value  $P$  in Table 7 to obtain the number of bits  $M$  in the  
787 Company Prefix and the number of digits  $L$  in the Company Prefix.
- 788 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned  
789 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a  
790 legal SGTIN-96 encoding. Otherwise, convert this integer into a decimal number  
791  $p_1p_2\dots p_L$ , adding leading zeros as necessary to make up  $L$  digits in total.
- 792 5. Extract the Item Reference and Indicator by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{38}$  as an  
793 unsigned integer. If this integer is greater than or equal to  $10^{(13-L)}$ , stop: the input bit  
794 string is not a legal SGTIN-96 encoding. Otherwise, convert this integer to a  $(13-L)$ -digit  
795 decimal number  $i_1i_2\dots i_{(13-L)}$ , adding leading zeros as necessary to make  $(13-L)$  digits.

- 796 6. Construct a 13-digit number  $d_1d_2\dots d_{13}$  where  $d_1 = i_1$  from Step 5,  $d_2d_3\dots d_{(L+1)} =$   
 797  $p_1p_2\dots p_L$  from Step 4, and  $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2 i_3\dots i_{(13-L)}$  from Step 5.
- 798 7. Calculate the check digit  $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 +$   
 799  $d_8 + d_{10} + d_{12})) \bmod 10$ .
- 800 8. The EAN.UCC GTIN-14 is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{14}$ .
- 801 9. Bits  $b_{37}b_{36}\dots b_0$ , considered as an unsigned integer, are the Serial Number.
- 802 10. (Optional) If it is desired to represent the serial number as a UCC/EAN-128  
 803 Application Identifier 21, convert the integer from Step 9 to a decimal string with no  
 804 leading zeros. If the integer in Step 9 is zero, convert it to a string consisting of the single  
 805 character "0".

806 **3.5 Serial Shipping Container Code (SSCC)**

807 The EPC encoding scheme for SSCC permits the direct embedding of EAN.UCC System  
 808 standard SSCC codes on EPC tags. In all cases, the check digit is not encoded. Two  
 809 encoding schemes are specified, SSCC-64 (64 bits) and SSCC-96 (96 bits).

810 In the 64-bit EPC, the limited number of bits prohibits a literal embedding of the  
 811 EAN.UCC Company Prefix. As a partial solution, a Company Prefix *Index* is used. This  
 812 Index, which can accommodate up to 16,384 codes, is assigned to companies that need to  
 813 use the 64 bit tags, in addition to their existing Company Prefixes. The Index is encoded  
 814 on the tag instead of the Company Prefix, and is subsequently translated to the Company  
 815 Prefix at low levels of the EPC system components (i.e. the Reader or Savant). While  
 816 this means a limited number of Company Prefixes can be represented in the 64-bit tag,  
 817 this is a transitional step to full accommodation in 96-bit and additional encoding  
 818 schemes.

819 **3.5.1 SSCC-64**

820 In addition to a Header, the EPC SSCC-64 is composed of three fields: the *Filter Value*,  
 821 *Company Prefix Index*, and *Serial Reference*, as shown in Table 8.

	Header	Filter Value	Company Prefix Index	Serial Reference
SSCC-64	8	3	14	39
	0000 1000 (Binary value)	(Refer to Table 9 for values)	16,383 (Max. decimal value)	99,999 - 99,999,999,999 (Max. decimal range*)

822 \*Max. decimal value range of Serial Reference field varies with the length of the Company Prefix

823 **Table 8.** The EPC 64-bit SSCC bit allocation, header, and maximum decimal values.

- 824 • *Header* is 8-bits, with a binary value of 0000 1000.

- 825 • *Filter Value* is not part of the SSCC or EPC identifier, but is used for fast filtering and  
 826 pre-selection of basic logistics types, such as cases and pallets. The Filter Values for  
 827 64-bit and 96-bit SSCC are the same. The normative specifications for Filter Values  
 828 are specified in Table 9. The value of 000 means “All Others”. That is, a filter value  
 829 of 000 means that the object to which the tag is affixed does not match any of the  
 830 logistic types defined as other filter values in this specification. It should be noted that  
 831 tags conforming to earlier versions of this specification, in which 000 was the only  
 832 value approved for use, will have filter value equal to 000 regardless of the logistic  
 833 types, but following the ratification of this standard, the filter value should be set to  
 834 match the object to which the tag is affixed, and use 000 only if the filter value for  
 835 such object does not exist in the specification.

Type	Binary Value
All Others	000
Undefined	001
Logistical / Shipping Unit	010
Reserved	011
Reserved	100
Reserved	101
Reserved	110
Reserved	111

836 **Table 9.** SSCC Filter Values

- 837 • *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this  
 838 field is not the Company Prefix itself, but rather an index into a table that provides the  
 839 Company Prefix as well as an indication of the Company Prefix’s length. The means  
 840 by which hardware or software may obtain the contents of the translation table is  
 841 specified in [Translation of 64-bit Tag Encoding Company Prefix Indices Into  
 842 EAN.UCC Company Prefixes].
- 843 • *Serial Reference* is a unique number for each instance, comprised of the Serial  
 844 Reference and the Extension digit. The Extension Digit is combined with the Serial  
 845 Reference field in the following manner: Leading zeros on the Serial Reference are  
 846 significant. Put the Extension Digit in the leftmost position available within the field.  
 847 *For instance, 000042235 is different than 42235. With the extension digit of 1, the*  
 848 *combination with 000042235 is 1000042235.* The resulting combination is treated as  
 849 a single integer, and encoded into binary to form the Serial Reference field. To avoid  
 850 unmanageably large and out-of-specification serial references, they should not exceed the  
 851 capacity specified in EAN.UCC specifications, which are (inclusive of extension digit)  
 852 9,999 for company prefixes of 12 digits up to 9,999,999,999 for company prefixes of 6  
 853 digits.

854 **3.5.1.1 SSCC-64 Encoding Procedure**

855 The following procedure creates an SSCC-64 encoding.

856 Given:

- 857 • An EAN.UCC SSCC consisting of digits  $d_1d_2\dots d_{18}$
- 858 • The length  $L$  of the company prefix portion of the SSCC
- 859 • A Filter Value  $F$  where  $0 \leq F < 8$

860 Procedure:

- 861 1. Extract the EAN.UCC Company Prefix  $d_2d_3\dots d_{(L+1)}$
- 862 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table  
863 to obtain the corresponding Company Prefix Index,  $C$ . If the Company Prefix was not  
864 found in the Company Prefix Translation Table, stop: this SSCC cannot be encoded in  
865 the SSCC-64 encoding.
- 866 3. Construct the Serial Reference + Extension Digit by concatenating digits  $d_1d_{(L+2)}$   
867  $d_{(L+3)}\dots d_{17}$  and considering the result to be a decimal integer,  $I$ . If  $I \geq 2^{39}$ , stop: this  
868 SSCC cannot be encoded in the SSCC-64 encoding.
- 869 4. Construct the final encoding by concatenating the following bit fields, from most  
870 significant to least significant: Header 00001000 (8 bits), Filter Value  $F$  (3 bits),  
871 Company Prefix Index  $C$  from Step 2 (14 bits), Serial Reference from Step 3 (39 bits).

872 **3.5.1.2 SSCC-64 Decoding Procedure**

873 Given:

- 874 • An SSCC-64 as a 64-bit bit string 00001000 $b_{55}b_{54}\dots b_0$  (where the first eight bits  
875 00001000 are the header)

876 Yields:

- 877 • An EAN.UCC SSCC
- 878 • A Filter Value

879 Procedure:

- 880 1. Bits  $b_{55}b_{54}b_{53}$ , considered as an unsigned integer, are the Filter Value.
- 881 2. Extract the Company Prefix Index  $C$  by considering bits  $b_{52}b_{51}\dots b_{39}$  as an unsigned  
882 integer.
- 883 3. Look up the Company Prefix Index  $C$  in the Company Prefix Translation Table to  
884 obtain the EAN.UCC Company Prefix  $p_1p_2\dots p_L$  consisting of  $L$  decimal digits (the value  
885 of  $L$  is also obtained from the table). If the Company Prefix Index  $C$  is not found in the  
886 Company Prefix Translation Table, stop: this bit string cannot be decoded as an SSCC-  
887 64.
- 888 4. Consider bits  $b_{38}b_{37}\dots b_0$  as an unsigned integer. If this integer is greater than or equal  
889 to  $10^{(17-L)}$ , stop: the input bit string is not a legal SSCC-64 encoding. Otherwise, convert



- 890 this integer to a (17-L)-digit decimal number  $i_1i_2\dots i_{(17-L)}$ , adding leading zeros as  
 891 necessary to make (17-L) digits.
- 892 5. Construct a 17-digit number  $d_1d_2\dots d_{17}$  where  $d_1 = s_1$  from Step 4,  $d_2d_3\dots d_{(L+1)} =$   
 893  $p_1p_2\dots p_L$  from Step 3, and  $d_{(L+2)}d_{(L+3)}\dots d_{17} = i_2 i_3\dots i_{(17-L)}$  from Step 4.
- 894 6. Calculate the check digit  $d_{18} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13} + d_{15} + d_{17}) - (d_2 +$   
 895  $d_4 + d_6 + d_8 + d_{10} + d_{12} + d_{14} + d_{16})) \bmod 10$ .
- 896 7. The EAN.UCC SSCC is the concatenation of digits from Steps 5 and 6:  $d_1d_2\dots d_{18}$ .

897 **3.5.2 SSCC-96**

898 In addition to a Header, the EPC SSCC-96 is composed of four fields: the *Filter Value*,  
 899 *Partition*, *Company Prefix*, and *Serial Reference*, as shown in Table 10.

	Header	Filter Value	Partition	Company Prefix	Serial Reference	Unallocated
SSCC-96	8	3	3	20-40	38-18	24
	0011 0001 (Binary value)	(Refer to Table 9 for values )	(Refer to Table 11 for values )	999,999 – 999,999,999,999 (Max. decimal range*)	99,999,999 – 99,999 (Max. decimal range*)	[Not Used]

900 \*Max. decimal value range of Company Prefix and Serial Reference fields vary according to the contents of  
 901 the Partition field.

902 **Table 10.** The EPC 96-bit SSCC bit allocation, header, and maximum decimal values.

- 903 • *Header* is 8-bits, with a binary value of 0011 0001.
- 904 • *Filter Value* is not part of the SSCC or EPC identifier, but is used for fast filtering and  
 905 pre-selection of basic logistics types. The Filter Values for 64-bit and 96-bit SSCC  
 906 are the same. See Table 9.
- 907 • The *Partition* is an indication of where the subsequent Company Prefix and Serial  
 908 Reference numbers are divided. This organization matches the structure in the  
 909 EAN.UCC SSCC in which the Company Prefix added to the Serial Reference number  
 910 (including the single Extension Digit) totals 17 digits, yet the Company Prefix may  
 911 vary from 6 to 12 digits and the Serial Reference from 11 to 5 digit(s). Table 11  
 912 shows allowed values of the partition value and the corresponding lengths of the  
 913 company prefix and serial reference.

914  
 915

Partition Value ( <i>P</i> )	Company Prefix		Serial Reference and Extension Digit	
	Bits ( <i>M</i> )	Digits ( <i>L</i> )	Bits ( <i>N</i> )	Digits
0	40	12	18	5
1	37	11	21	6
2	34	10	24	7
3	30	9	28	8
4	27	8	31	9
5	24	7	34	10
6	20	6	38	11

**Table 11.** SSCC-96 Partitions.

916

917

- *Company Prefix* contains a literal embedding of the Company Prefix.

918

919

920

921

922

923

924

925

926

927

928

- *Serial Reference* is a unique number for each instance, comprised of the Serial Reference and the Extension digit. The Extension Digit is combined with the Serial Reference field in the following manner: Leading zeros on the Serial Reference are significant. Put the Extension Digit in the leftmost position available within the field. For instance, 000042235 is different than 42235. With the extension digit of 1, the combination with 000042235 is 1000042235. The resulting combination is treated as a single integer, and encoded into binary to form the Serial Reference field. To avoid unmanageably large and out-of-specification serial references, they should not exceed the capacity specified in EAN.UCC specifications, which are (inclusive of extension digit) 9,999 for company prefixes of 12 digits up to 9,999,999,999 for company prefixes of 6 digits.

929

930

- *Unallocated* is not used. This field must contain zeros to conform with this version of the specification.

931

### 3.5.2.1 SSCC-96 Encoding Procedure

932

The following procedure creates an SSCC-96 encoding.

933

Given:

934

935

936

- An EAN.UCC SSCC consisting of digits  $d_1d_2\dots d_{18}$
- The length  $L$  of the Company Prefix portion of the SSCC
- A Filter Value  $F$  where  $0 \leq F < 8$

937

Procedure:

938

939

1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of the Partition Table (Table 11) to determine the Partition Value,  $P$ , the number of bits

940  $M$  in the Company Prefix field, and the number of bits  $N$  in the Serial Reference and  
941 Extension Digit field. If  $L$  is not found in any row of Table 11, stop: this SSCC cannot  
942 be encoded in an SSCC-96.

943 2. Construct the Company Prefix by concatenating digits  $d_2d_3\dots d_{(L+1)}$  and considering  
944 the result to be a decimal integer,  $C$ .

945 3. Construct the Serial Reference + Extension Digit by concatenating digits  
946  $d_1d_{(L+2)}d_{(L+3)}\dots d_{17}$  and considering the result to be a decimal integer,  $S$ .

947 4. Construct the final encoding by concatenating the following bit fields, from most  
948 significant to least significant: Header 00110001 (8 bits), Filter Value  $F$  (3 bits),  
949 Partition Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Serial  
950 Reference  $S$  from Step 3 ( $N$  bits), and 24 zero bits. Note that  $M+N = 58$  bits for all  $P$ .

### 951 **3.5.2.2 SSCC-96 Decoding Procedure**

952 Given:

- 953 • An SSCC-96 as a 96-bit bit string 00110001 $b_{87}b_{86}\dots b_0$  (where the first eight bits  
954 00110001 are the header)

955 Yields:

- 956 • An EAN.UCC SSCC
- 957 • A Filter Value

958 Procedure:

959 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.

960 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
961  $P = 7$ , stop: this bit string cannot be decoded as an SSCC-96.

962 3. Look up the Partition Value  $P$  in Table 11 to obtain the number of bits  $M$  in the  
963 Company Prefix and the number of digits  $L$  in the Company Prefix.

964 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned  
965 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a  
966 legal SSCC-96 encoding. Otherwise, convert this integer into a decimal number  
967  $p_1p_2\dots p_L$ , adding leading zeros as necessary to make up  $L$  digits in total.

968 5. Extract the Serial Reference by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{24}$  as an unsigned  
969 integer. If this integer is greater than or equal to  $10^{(17-L)}$ , stop: the input bit string is not a  
970 legal SSCC-96 encoding. Otherwise, convert this integer to a  $(17-L)$ -digit decimal  
971 number  $i_1i_2\dots i_{(17-L)}$ , adding leading zeros as necessary to make  $(17-L)$  digits.

972 6. Construct a 17-digit number  $d_1d_2\dots d_{17}$  where  $d_1 = s_1$  from Step 5,  $d_2d_3\dots d_{(L+1)} =$   
973  $p_1p_2\dots p_L$  from Step 4, and  $d_{(L+2)}d_{(L+3)}\dots d_{17} = i_2i_3\dots i_{(17-L)}$  from Step 5.

974 7. Calculate the check digit  $d_{18} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13} + d_{15} + d_{17}) - (d_2 +$   
975  $d_4 + d_6 + d_8 + d_{10} + d_{12} + d_{14} + d_{16})) \bmod 10$ .

976 8. The EAN.UCC SSCC is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{18}$ .

977 **3.6 Serialized Global Location Number (SGLN)**

978 The EPC encoding scheme for GLN permits the direct embedding of EAN.UCC System  
 979 standard GLN on EPC tags. The serial number field is not used. In all cases the check  
 980 digit is not encoded. Two encoding schemes are specified, SGLN-64 (64 bits) and  
 981 SGLN-96 (96 bits).

982 In the SGLN-64 encoding, the limited number of bits prohibits a literal embedding of the  
 983 GLN. As a partial solution, a Company Prefix *Index* is used. This *index*, which can  
 984 accommodate up to 16,384 codes, is assigned to companies that need to use the 64 bit  
 985 tags, in addition to their existing EAN.UCC Company Prefixes. The *index* is encoded on  
 986 the tag instead of the Company Prefix, and is subsequently translated to the Company  
 987 Prefix at low levels of the EPC system components (i.e. the Reader or Savant).

988 While this means a limited number of Company Prefixes can be represented in the 64-bit  
 989 tag, this is a transitional step to full accommodation in 96-bit and additional encoding  
 990 schemes.

991 **3.6.1 SGLN-64**

992 The SGLN-64 includes *four* fields in addition to the header – *Filter Value*, *Company*  
 993 *Prefix Index*, *Location Reference*, and *Serial Number*, as shown in Table 12.

994

	Header	Filter Value	Company Prefix Index	Location Reference	Serial Number
SGLN-64	8	3	14	20	19
	0000 1001 (Binary value)	(Refer to Table 13 for values )	16,383 (Max. decimal value)	999,999 - 0 (Max. decimal range*)	524,288 (Max. decimal value) [Not Used]

995 \*Max. decimal value range of Location Reference field varies with the length of the Company Prefix

996 **Table 12.** The EPC SGLN-64 bit allocation, header, and maximum decimal values.

997

- 998
- *Header* is 8 bits, with a binary value of 0000 1001.

999

  - *Filter Value* is not part of the SGLN pure identity, but is additional data that is used  
 1000 for fast filtering and pre-selection of basic location types. The Filter Values for 64-bit  
 1001 and 96-bit SGLN are the same. See Table 13 for currently defined filter values.

1002

  - *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this  
 1003 field is not the Company Prefix itself, but rather an index into a table that provides the

1004 Company Prefix as well as an indication of the Company Prefix's length. The means  
 1005 by which hardware or software may obtain the contents of the translation table is  
 1006 specified in [Translation of 64-bit Tag Encoding Company Prefix Indices Into  
 1007 EAN.UCC Company Prefixes].

- 1008 • *Location Reference* encodes the GLN Location Reference number.
- 1009 • *Serial Number* contains a serial number. Note: The serial number field is reserved and  
 1010 should not be used, until the EAN.UCC community determines the appropriate way,  
 1011 if any, for extending GLN.

1012

Type	Binary Value
All Others	000
Reserved	001
Reserved	010
Reserved	011
Reserved	100
Reserved	101
Reserved	110
Reserved	111

1013 **Table 13.** SGLN Filter Values .

1014 **3.6.1.1 SGLN-64 Encoding Procedure**

1015 The following procedure creates an SGLN-64 encoding.

1016 Given:

- 1017 • An EAN.UCC GLN consisting of digits  $d_1d_2\dots d_{13}$
- 1018 • The length  $L$  of the company prefix portion of the GLN
- 1019 • A Serial Number  $S$  where  $0 \leq S < 2^{19}$
- 1020 • A Filter Value  $F$  where  $0 \leq F < 8$

1021 Procedure:

- 1022 1. Extract the EAN.UCC Company Prefix  $d_1d_2\dots d_L$
- 1023 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table  
 1024 to obtain the corresponding Company Prefix Index,  $C$ . If the Company Prefix was not  
 1025 found in the Company Prefix Translation Table, stop: this GLN cannot be encoded in the  
 1026 SGLN-64 encoding.

- 1027 3. Construct the Location Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and  
 1028 considering the result to be a decimal integer,  $I$ . If  $I \geq 2^{20}$ , stop: this GLN cannot be  
 1029 encoded in the SGLN-64 encoding.
- 1030 4. Construct the final encoding by concatenating the following bit fields, from most  
 1031 significant to least significant: Header 00001001 (8 bits), Filter Value  $F$  (3 bits),  
 1032 Company Prefix Index  $C$  from Step 2 (14 bits), Location Reference from Step 3 (20 bits),  
 1033 Serial Number  $S$  (19 bits).

### 1034 **3.6.1.2 SGLN-64 Decoding Procedure**

1035 Given:

- 1036 • An SGLN-64 as a 64-bit bit string  $00001001b_{55}b_{54}\dots b_0$  (where the first eight bits  
 1037  $00001001$  are the header)

1038 Yields:

- 1039 • An EAN.UCC GLN  
 1040 • A Serial Number  
 1041 • A Filter Value

1042 Procedure:

- 1043 1. Bits  $b_{55}b_{54}b_{53}$ , considered as an unsigned integer, are the Filter Value.
- 1044 2. Extract the Company Prefix Index  $C$  by considering bits  $b_{52}b_{51}\dots b_{39}$  as an unsigned  
 1045 integer.
- 1046 3. Look up the Company Prefix Index  $C$  in the Company Prefix Translation Table to  
 1047 obtain the EAN.UCC Company Prefix  $p_1p_2\dots p_L$  consisting of  $L$  decimal digits (the value  
 1048 of  $L$  is also obtained from the table). If the Company Prefix Index  $C$  is not found in the  
 1049 Company Prefix Translation Table, stop: this bit string cannot be decoded as an SGLN-  
 1050 64.
- 1051 4. Consider bits  $b_{38}b_{37}\dots b_{19}$  as an unsigned integer. If this integer is greater than or  
 1052 equal to  $10^{(12-L)}$ , stop: the input bit string is not a legal SGLN-64 encoding. Otherwise,  
 1053 convert this integer to a  $(12-L)$ -digit decimal number  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as  
 1054 necessary to make  $(12-L)$  digits.
- 1055 5. Construct a 12-digit number  $d_1d_2\dots d_{12}$  where  $d_1d_2\dots d_L = p_1p_2\dots p_L$  from Step 3, and  
 1056  $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_1i_2\dots i_{(12-L)}$  from Step 4.
- 1057 6. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 +$   
 1058  $d_9 + d_{11})) \bmod 10$ .
- 1059 7. The EAN.UCC GLN is the concatenation of digits from Steps 5 and 6:  $d_1d_2\dots d_{13}$ .
- 1060 8. Bits  $b_{18}b_{17}\dots b_0$ , considered as an unsigned integer, are the Serial Number.

1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075

### 3.6.2 SGLN-96

In addition to a Header, the SGLN-96 is composed of five fields: the *Filter Value*, *Partition*, *Company Prefix*, *Location Reference*, and *Serial Number*, as shown in Table 14.

- *Header* is 8-bits, with a binary value of 0011 0010.
- *Filter Value* is not part of the GLN or EPC identifier, but is used for fast filtering and pre-selection of basic location types. The Filter Values for 64-bit and 96-bit GLN are the same. See Table 13.
- *Partition* is an indication of where the subsequent Company Prefix and Location Reference numbers are divided. This organization matches the structure in the EAN.UCC GLN in which the Company Prefix added to the Location Reference number totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the Location Reference number from 6 to 0 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Location Reference* fields are defined in Table 15.

	Header	Filter Value	Partition	Company Prefix	Location Reference	Serial Number
SGLN-96	8	3	3	20-40	21-1	41
	0011 0010 (Binary value)	(Refer to Table 13 for values )	(Refer to Table 15 for values )	999,999 – 999,999,999,999 (Max. decimal range*)	999,999 – 0 (Max. decimal range*)	2,199,023,255,551 (Max. decimal value) [Not Used]

1076  
1077

\*Max. decimal value range of Company Prefix and Location Reference fields vary according to contents of the Partition field.

1078  
1079

**Table 14.** The EPC SGLN-96 bit allocation, header, and maximum decimal values.

1080  
1081  
1082  
1083  
1084  
1085

- *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.
- *Location Reference* encodes the GLN Location Reference number.
- *Serial Number* contains a serial number. Note: The serial number field is reserved and should not be used, until the EAN.UCC community determines the appropriate way, if any, for extending GLN.

Partition Value ( $P$ )	Company Prefix		Location Reference	
	Bits ( $M$ )	Digits ( $L$ )	Bits ( $N$ )	Digits
0	40	12	1	0
1	37	11	4	1
2	34	10	7	2
3	30	9	11	3
4	27	8	14	4
5	24	7	17	5
6	20	6	21	6

1086

**Table 15.** SGLN-96 Partitions.

1087

### 3.6.2.1 SGLN-96 Encoding Procedure

1088

The following procedure creates an SGLN-96 encoding.

1089

Given:

1090

- An EAN.UCC GLN consisting of digits  $d_1d_2\dots d_{13}$

1091

- The length  $L$  of the Company Prefix portion of the GLN

1092

- A Serial Number  $S$  where  $0 \leq S < 2^{41}$

1093

- A Filter Value  $F$  where  $0 \leq F < 8$

1094

Procedure:

1095

1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of the Partition Table (Table 15) to determine the Partition Value,  $P$ , the number of bits  $M$  in the Company Prefix field, and the number of bits  $N$  in the Location Reference field.

1096

1097

If  $L$  is not found in any row of Table 15, stop: this GLN cannot be encoded in an SGLN-96.

1098

1100

2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_L$  and considering the result to be a decimal integer,  $C$ .

1101

1102

3. Construct the Location Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and considering the result to be a decimal integer,  $I$ .

1103

1104

4. Construct the final encoding by concatenating the following bit fields, from most significant to least significant: Header 00110010 (8 bits), Filter Value  $F$  (3 bits), Partition Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Location Reference from Step 3 ( $N$  bits), Serial Number  $S$  (41 bits). Note that  $M+N = 41$  bits for all  $P$ .

1105

1106

1107

1108



### 1109 3.6.2.2 SGLN-96 Decoding Procedure

1110 Given:

- 1111 • An SGLN-96 as a 96-bit bit string  $00110010b_{87}b_{86}\dots b_0$  (where the first eight bits
- 1112  $00110010$  are the header)

1113 Yields:

- 1114 • An EAN.UCC GLN
- 1115 • A Serial Number
- 1116 • A Filter Value

1117 Procedure:

- 1118 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1119 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If
- 1120  $P = 7$ , stop: this bit string cannot be decoded as an SGLN-96.
- 1121 3. Look up the Partition Value  $P$  in Table 15 to obtain the number of bits  $M$  in the
- 1122 Company Prefix and the number of digits  $L$  in the Company Prefix.
- 1123 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned
- 1124 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a
- 1125 legal SGLN-96 encoding. Otherwise, convert this integer into a decimal number
- 1126  $p_1p_2\dots p_L$ , adding leading zeros as necessary to make up  $L$  digits in total.
- 1127 5. Extract the Location Reference by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{41}$  as an unsigned
- 1128 integer. If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit string is not a
- 1129 legal SGLN-96 encoding. Otherwise, convert this integer to a  $(12-L)$ -digit decimal
- 1130 number  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as necessary to make  $(12-L)$  digits.
- 1131 6. Construct a 12-digit number  $d_1d_2\dots d_{12}$  where  $d_1d_2\dots d_L = p_1p_2\dots p_L$  from Step 4, and
- 1132  $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_2i_3\dots i_{(12-L)}$  from Step 5.
- 1133 7. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 +$
- 1134  $d_9 + d_{11})) \bmod 10$ .
- 1135 8. The EAN.UCC GLN is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{13}$ .
- 1136 9. Bits  $b_{40}b_{39}\dots b_0$ , considered as an unsigned integer, are the Serial Number.

### 1137 3.7 Global Returnable Asset Identifier (GRAI)

1138 The EPC encoding scheme for GRAI permits the direct embedding of EAN.UCC System  
1139 standard GRAI on EPC tags. In all cases, the check digit is not encoded. Two encoding  
1140 schemes are specified, GRAI-64 (64 bits) and GRAI-96 (96 bits).

1141 In the GRAI-64 encoding, the limited number of bits prohibits a literal embedding of the  
1142 GRAI. As a partial solution, a Company Prefix *Index* is used. This Index, which can  
1143 accommodate up to 16,384 codes, is assigned to companies that need to use the 64 bit  
1144 tags, in addition to their existing EAN.UCC Company Prefixes. The Index is encoded on  
1145 the tag instead of the Company Prefix, and is subsequently translated to the Company

1146 Prefix at low levels of the EPC system components (i.e. the Reader or Savant). While  
 1147 this means that only a limited number of Company Prefixes can be represented in the 64-  
 1148 bit tag, this is a transitional step to full accommodation in 96-bit and additional encoding  
 1149 schemes.

1150 **3.7.1 GRAI-64**

1151 The GRAI-64 includes *four* fields in addition to the Header – *Filter Value, Company*  
 1152 *Prefix Index, Asset Type, and Serial Number*, as shown in Table 16.

1153

	Header	Filter Value	Company Prefix Index	Asset Type	Serial Number
GRAI-64	8	3	14	20	19
	0000 1010 (Binary value)	(Refer to Table 17 for values )	16,383 (Max. decimal value)	999,999 - 0 (Max. decimal range*)	524,287 (Max. decimal value)

1154

\*Max. decimal value range of Asset Type field varies with Company Prefix.

1155

**Table 16.** The EPC GRAI-64 bit allocation, header, and maximum decimal values.

1156

- 1157 • *Header* is 8 bits, with a binary value of 0000 1010.
- 1158 • *Filter Value* is not part of the GRAI pure identity, but is additional data that is used  
 1159 for fast filtering and pre-selection of basic asset types. The Filter Values for 64-bit  
 1160 and 96-bit GRAI are the same. See Table 17 for currently defined GRAI filter values.  
 1161 This specification anticipates that valuable Filter Values will be determined once  
 1162 there has been time to consider the possible use cases.

Type	Binary Value
All Others	000
Reserved	001
Reserved	010
Reserved	011
Reserved	100
Reserved	101

Type	Binary Value
Reserved	110
Reserved	111

1163

1164

**Table 17.** GRAI Filter Values

- 1165 • *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this  
1166 field is not the Company Prefix itself, but rather an index into a table that provides the  
1167 Company Prefix as well as an indication of the Company Prefix's length. The means  
1168 by which hardware or software may obtain the contents of the translation table is  
1169 specified in [Translation of 64-bit Tag Encoding Company Prefix Indices Into  
1170 EAN.UCC Company Prefixes].
- 1171 • *Asset Type* encodes the GRAI Asset Type number.
- 1172 • *Serial Number* contains a serial number. The 64-bit and 96-bit tag encodings are only  
1173 capable of representing a subset of Serial Numbers allowed in the General EAN.UCC  
1174 Specifications. The capacity of this mandatory serial number is less than the  
1175 maximum EAN.UCC System specification for serial number, no leading zeros are  
1176 permitted, and only numbers are permitted.

### 1177 3.7.1.1 GRAI-64 Encoding Procedure

1178 The following procedure creates a GRAI-64 encoding.

1179 Given:

- 1180 • An EAN.UCC GRAI consisting of digits  $0d_2\dots d_K$ , where  $15 \leq K \leq 30$ .
- 1181 • The length  $L$  of the company prefix portion of the GRAI
- 1182 • A Filter Value  $F$  where  $0 \leq F < 8$

1183 Procedure:

- 1184 1. Extract the EAN.UCC Company Prefix  $d_2d_3\dots d_{L+1}$
- 1185 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table  
1186 to obtain the corresponding Company Prefix Index,  $C$ . If the Company Prefix was not  
1187 found in the Company Prefix Translation Table, stop: this GRAI cannot be encoded in  
1188 the GRAI-64 encoding.
- 1189 3. Construct the Asset Type by concatenating digits  $d_{(L+2)}d_{(L+3)}\dots d_{13}$  and considering the  
1190 result to be a decimal integer,  $I$ . If  $I \geq 2^{20}$ , stop: this GRAI cannot be encoded in the  
1191 GRAI-64 encoding.
- 1192 4. Construct the Serial Number by concatenating digits  $d_{15}d_{16}\dots d_K$ . If any of these  
1193 characters is not a digit, stop: this GRAI cannot be encoded in the GRAI-64 encoding.  
1194 Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^{19}$ , stop: this GRAI  
1195 cannot be encoded in the GRAI-64 encoding. Also, if  $K > 15$  and  $d_{15} = 0$ , stop: this

1196 GRAI cannot be encoded in the GRAI-64 encoding (because leading zeros are not  
 1197 permitted except in the case where the Serial Number consists of a single zero digit).  
 1198 5. Construct the final encoding by concatenating the following bit fields, from most  
 1199 significant to least significant: Header 00001010 (8 bits), Filter Value  $F$  (3 bits),  
 1200 Company Prefix Index  $C$  from Step 2 (14 bits), Asset Type  $I$  from Step 3 (20 bits), Serial  
 1201 Number  $S$  from Step 4 (19 bits).

### 1202 **3.7.1.2 GRAI-64 Decoding Procedure**

1203 Given:

- 1204 • An GRAI-64 as a 64-bit bit string 00001010 $b_{55}b_{54}\dots b_0$  (where the first eight bits  
 1205 00001010 are the header)

1206 Yields:

- 1207 • An EAN.UCC GRAI
- 1208 • A Filter Value

1209 Procedure:

- 1210 1. Bits  $b_{55}b_{54}b_{53}$ , considered as an unsigned integer, are the Filter Value.
- 1211 2. Extract the Company Prefix Index  $C$  by considering bits  $b_{52}b_{51}\dots b_{39}$  as an unsigned  
 1212 integer.
- 1213 3. Look up the Company Prefix Index  $C$  in the Company Prefix Translation Table to  
 1214 obtain the EAN.UCC Company Prefix  $p_1p_2\dots p_L$  consisting of  $L$  decimal digits (the value  
 1215 of  $L$  is also obtained from the table). If the Company Prefix Index  $C$  is not found in the  
 1216 Company Prefix Translation Table, stop: this bit string cannot be decoded as a GRAI-64.
- 1217 4. Consider bits  $b_{38}b_{37}\dots b_{19}$  as an unsigned integer. If this integer is greater than or  
 1218 equal to  $10^{(12-L)}$ , stop: the input bit string is not a legal GRAI-64 encoding. Otherwise,  
 1219 convert this integer to a  $(12-L)$ -digit decimal number  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as  
 1220 necessary to make  $(12-L)$  digits.
- 1221 5. Construct a 13-digit number  $0d_2d_3\dots d_{13}$  where  $d_2d_3\dots d_{L+1} = p_1p_2\dots p_L$  from Step 3, and  
 1222  $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_1i_2\dots i_{(12-L)}$  from Step 4.
- 1223 6. Calculate the check digit  $d_{14} = (-3(d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8 +$   
 1224  $d_{10} + d_{12})) \bmod 10$ .
- 1225 7. Consider bits  $b_{18}b_{17}\dots b_0$  as an unsigned integer. Convert this integer into a decimal  
 1226 number  $d_{15}d_{16}\dots d_K$ , with no leading zeros (exception: if the integer is equal to zero,  
 1227 convert it to a single zero digit).
- 1228 8. The EAN.UCC GRAI is the concatenation of the digits from Steps 5, 6, and 7:  
 1229  $0d_2d_3\dots d_K$ .

### 1230 **3.7.2 GRAI-96**

1231 In addition to a Header, the GRAI-96 is composed of five fields: the *Filter Value*,  
 1232 *Partition*, *Company Prefix*, *Asset Type*, and *Serial Number*, as shown in Table 18.

	Header	Filter Value	Partition	Company Prefix	Asset Type	Serial Number
GRAI-96	8	3	3	20-40	24-4	38
	0011 0011 (Binary value)	(Refer to Table 17 for values )	(Refer to Table 19 for values )	999,999 – 999,999,999,999 (Max. decimal range*)	999,999 – 0 (Max. decimal range*)	274,877,906,943 (Max. decimal value)

1233  
1234

\*Max. decimal value range of Company Prefix and Asset Type fields vary according to contents of the Partition field.

1235

**Table 18.** The EPC GRAI-96 bit allocation, header, and maximum decimal values.

1236

- *Header* is 8-bits, with a binary value of 0011 0011.

1237

- *Filter Value* is not part of the GRAI or EPC identifier, but is used for fast filtering and pre-selection of basic asset types. The Filter Values for 64-bit and 96-bit GRAI are the same. See Table 17.

1238  
1239

1240

- *Partition* is an indication of where the subsequent Company Prefix and Asset Type numbers are divided. This organization matches the structure in the EAN.UCC GRAI in which the Company Prefix added to the Asset Type number totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the Asset Type from 6 to 0 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Asset Type* fields are defined in Table 19.

1241  
1242  
1243  
1244  
1245

Partition Value (P)	Company Prefix		Asset Type	
	Bits (M)	Digits (L)	Bits (N)	Digits
0	40	12	4	0
1	37	11	7	1
2	34	10	10	2
3	30	9	14	3
4	27	8	17	4
5	24	7	20	5
6	20	6	24	6

1246

**Table 19.** GRAI-96 Partitions.

1247

- 1248 • *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.
- 1249 • *Asset Type* encodes the GRAI Asset Type number.
- 1250 • *Serial Number* contains a serial number. The 64-bit and 96-bit tag encodings are only  
1251 capable of representing a subset of Serial Numbers allowed in the General EAN.UCC  
1252 Specifications. The capacity of this mandatory serial number is less than the  
1253 maximum EAN.UCC System specification for serial number, no leading zeros are  
1254 permitted, and only numbers are permitted.

### 1255 **3.7.2.1 GRAI-96 Encoding Procedure**

1256 The following procedure creates a GRAI-96 encoding.

1257 Given:

- 1258 • An EAN.UCC GRAI consisting of digits  $0d_2d_3\dots d_K$ , where  $15 \leq K \leq 30$ .
- 1259 • The length  $L$  of the Company Prefix portion of the GRAI
- 1260 • A Filter Value  $F$  where  $0 \leq F < 8$

1261 Procedure:

- 1262 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column  
1263 of the Partition Table (Table 19) to determine the Partition Value,  $P$ , the number of bits  
1264  $M$  in the Company Prefix field, and the number of bits  $N$  in Asset Type field. If  $L$  is not  
1265 found in any row of Table 19, stop: this GRAI cannot be encoded in a GRAI-96.
- 1266 2. Construct the Company Prefix by concatenating digits  $d_2d_3\dots d_{(L+1)}$  and considering  
1267 the result to be a decimal integer,  $C$ .
- 1268 3. Construct the Asset Type by concatenating digits  $d_{(L+2)}d_{(L+3)}\dots d_{13}$  and considering the  
1269 result to be a decimal integer,  $I$ .
- 1270 4. Construct the Serial Number by concatenating digits  $d_{15}d_{16}\dots d_K$ . If any of these  
1271 characters is not a digit, stop: this GRAI cannot be encoded in the GRAI-96 encoding.  
1272 Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^{38}$ , stop: this GRAI  
1273 cannot be encoded in the GRAI-96 encoding. Also, if  $K > 15$  and  $d_{15} = 0$ , stop: this  
1274 GRAI cannot be encoded in the GRAI-96 encoding (because leading zeros are not  
1275 permitted except in the case where the Serial Number consists of a single zero digit).
- 1276 5. Construct the final encoding by concatenating the following bit fields, from most  
1277 significant to least significant: Header 00110011 (8 bits), Filter Value  $F$  (3 bits),  
1278 Partition Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Asset  
1279 Type  $I$  from Step 3 ( $N$  bits), Serial Number  $S$  from Step 4 (38 bits). Note that  $M+N =$   
1280 44 bits for all  $P$ .

### 1281 **3.7.2.2 GRAI-96 Decoding Procedure**

1282 Given:

- 1283 • An GRAI-96 as a 96-bit bit string  $00110011b_{87}b_{86}\dots b_0$  (where the first eight bits  
1284  $00110011$  are the header)
- 1285 Yields:
- 1286 • An EAN.UCC GRAI
- 1287 • A Filter Value
- 1288 Procedure:
- 1289 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1290 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
1291  $P = 7$ , stop: this bit string cannot be decoded as a GRAI-96.
- 1292 3. Look up the Partition Value  $P$  in Table 19 to obtain the number of bits  $M$  in the  
1293 Company Prefix and the number of digits  $L$  in the Company Prefix.
- 1294 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned  
1295 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a  
1296 legal GRAI-96 encoding. Otherwise, convert this integer into a decimal number  
1297  $p_1p_2\dots p_L$ , adding leading zeros as necessary to make up  $L$  digits in total.
- 1298 5. Extract the Asset Type by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{38}$  as an unsigned integer.  
1299 If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit string is not a legal  
1300 GRAI-96 encoding. Otherwise, convert this integer to a  $(12-L)$ -digit decimal number  
1301  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as necessary to make  $(12-L)$  digits.
- 1302 6. Construct a 13-digit number  $0d_2d_3\dots d_{13}$  where  $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$  from Step 4,  
1303 and  $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_1i_2\dots i_{(12-L)}$  from Step 5.
- 1304 7. Calculate the check digit  $d_{14} = (-(-3(d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8$   
1305  $+ d_{10} + d_{12}))) \bmod 10$ .
- 1306 8. Extract the Serial Number by considering bits  $b_{37}b_{36}\dots b_0$  as an unsigned integer.  
1307 Convert this integer to a decimal number  $d_{15}d_{16}\dots d_K$ , with no leading zeros (exception: if  
1308 the integer is equal to zero, convert it to a single zero digit).
- 1309 9. The EAN.UCC GRAI is the concatenation of a single zero digit and the digits from  
1310 Steps 6, 7, and 8:  $0d_2d_3\dots d_K$ .

### 1311 **3.8 Global Individual Asset Identifier (GIAI)**

1312 The EPC encoding scheme for GIAI permits the direct embedding of EAN.UCC System  
1313 standard GIAI codes on EPC tags (except as noted below for 64-bit tags). Two encoding  
1314 schemes are specified, GIAI-64 (64 bits) and GIAI-96 (96 bits).

1315 In the 64-bit EPC, the limited number of bits prohibits a literal embedding of the  
1316 EAN.UCC Company Prefix. As a partial solution, a Company Prefix *Index* is used. In  
1317 addition to their existing Company Prefixes, this Index, which can accommodate up to  
1318 16,384 codes, is assigned to companies that need to use the 64 bit tags. The Index is  
1319 encoded on the tag instead of the Company Prefix, and is subsequently translated to the  
1320 Company Prefix at low levels of the EPC system components (i.e. the Reader or Savant).

1321 While this means a limited number of Company Prefixes can be represented in the 64-bit  
 1322 tag, this is a transitional step to full accommodation in 96-bit and additional encoding  
 1323 schemes.

1324 **3.8.1 GIAI-64**

1325 In addition to a Header, the EPC GIAI-64 is composed of three fields: the *Filter Value*,  
 1326 *Company Prefix Index*, and *Individual Asset Reference*, as shown in Table 20.

1327

	Header	Filter Value	Company Prefix Index	Individual Asset Reference
GIAI-64	8	3	14	39
	0000 1011 (Binary value)	(Refer to Table 21 for values )	16,383 (Max. decimal value)	549,755,813,887 (Max. decimal value)

1328 **Table 20.** The EPC 64-bit GIAI bit allocation, header, and maximum decimal values.

- 1329
- *Header* is 8-bits, with a binary value of 0000 1011.
  - *Filter Value* is not part of the GIAI pure identity, but is additional data that is used for fast filtering and pre-selection of basic asset types. The Filter Values for 64-bit and 96-bit GIAI are the same. See Table 21 for currently defined GIAI filter values. This specification anticipates that valuable Filter Values will be determined once there has been time to consider the possible use cases.

1335

Type	Binary Value
All Others	000
Reserved	001
Reserved	010
Reserved	011
Reserved	100
Reserved	101
Reserved	110
Reserved	111

1336

**Table 21.** GIAI Filter Values



- 1337 • *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this  
 1338 field is not the Company Prefix itself, but rather an index into a table that provides the  
 1339 Company Prefix as well as an indication of the Company Prefix's length. The means  
 1340 by which hardware or software may obtain the contents of the translation table is  
 1341 specified in [Translation of 64-bit Tag Encoding Company Prefix Indices Into  
 1342 EAN.UCC Company Prefixes].
- 1343 • *Individual Asset Reference* is a unique number for each instance. The 64-bit and 96-  
 1344 bit tag encodings are only capable of representing a subset of asset references allowed  
 1345 in the General EAN.UCC Specifications. The capacity of this asset reference is less  
 1346 than the maximum EAN.UCC System specification for asset references, no leading  
 1347 zeros are permitted, and only numbers are permitted.

1348 **3.8.1.1 GIAI-64 Encoding Procedure**

1349 The following procedure creates a GIAI-64 encoding.

1350 Given:

1351 An EAN.UCC GIAI consisting of digits  $d_1d_2\dots d_K$  where  $K \leq 30$ .

1352 The length  $L$  of the company prefix portion of the GIAI

1353 A Filter Value  $F$  where  $0 \leq F < 8$

1354 Procedure:

- 1355 1. Extract the EAN.UCC Company Prefix  $d_1d_2\dots d_L$
- 1356 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table  
 1357 to obtain the corresponding Company Prefix Index,  $C$ . If the Company Prefix was not  
 1358 found in the Company Prefix Translation Table, stop: this GIAI cannot be encoded in the  
 1359 GIAI-64 encoding.
- 1360 3. Construct the Individual Asset Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_K$ . If  
 1361 any of these characters is not a digit, stop: this GIAI cannot be encoded in the GIAI-64  
 1362 encoding. Otherwise, consider the result to be a decimal integer,  $I$ . If  $I \geq 2^{39}$ , stop: this  
 1363 GIAI cannot be encoded in the GIAI-64 encoding. Also, if  $K > L+1$  and  $d_{(L+1)} = 0$ , stop:  
 1364 this GIAI cannot be encoded in the GIAI-64 encoding (because leading zeros are not  
 1365 permitted except in the case where the Individual Asset Reference consists of a single  
 1366 zero digit).
- 1367 4. Construct the final encoding by concatenating the following bit fields, from most  
 1368 significant to least significant: Header 00001011 (8 bits), Filter Value  $F$  (3 bits),  
 1369 Company Prefix Index  $C$  from Step 2 (14 bits), Individual Asset Reference from Step 3  
 1370 (39 bits).

1371 **3.8.1.2 GIAI-64 Decoding Procedure**

1372 Given:

1373 An GIAI-64 as a 64-bit bit string 00001011 $b_{55}b_{54}\dots b_0$  (where the first eight bits  
 1374 00001011 are the header)

- 1375 Yields:
- 1376 An EAN.UCC GIAI
- 1377 A Filter Value
- 1378 Procedure:
- 1379 1. Bits  $b_{55}b_{54}b_{53}$ , considered as an unsigned integer, are the Filter Value.
- 1380 2. Extract the Company Prefix Index  $C$  by considering bits  $b_{52}b_{51}\dots b_{39}$  as an unsigned
- 1381 integer.
- 1382 3. Look up the Company Prefix Index  $C$  in the Company Prefix Translation Table to
- 1383 obtain the EAN.UCC Company Prefix  $p_1p_2\dots p_L$  consisting of  $L$  decimal digits (the value
- 1384 of  $L$  is also obtained from the table). If the Company Prefix Index  $C$  is not found in the
- 1385 Company Prefix Translation Table, stop: this bit string cannot be decoded as a GIAI-64.
- 1386 4. Consider bits  $b_{38}b_{37}\dots b_0$  as an unsigned integer. If this integer is greater than or equal
- 1387 to  $10^{(30-L)}$ , stop: the input bit string is not a legal GIAI-64 encoding. Otherwise, convert
- 1388 this integer to a decimal number  $s_1s_2\dots s_J$ , with no leading zeros (exception: if the integer
- 1389 is equal to zero, convert it to a single zero digit).
- 1390 5. Construct a  $K$ -digit number  $d_1d_2\dots d_K$  where  $d_1d_2\dots d_L = p_1p_2\dots p_L$  from Step 3, and
- 1391  $d_{(L+1)}d_{(L+2)}\dots d_K = s_1s_2\dots s_J$  from Step 4. This  $K$ -digit number, where  $K \leq 30$ , is the
- 1392 EAN.UCC GIAI.

1393 **3.8.2 GIAI-96**

1394 In addition to a Header, the EPC GIAI-96 is composed of four fields: the *Filter Value*,

1395 *Partition*, *Company Prefix*, and *Individual Asset Reference*, as shown in Table 22.

1396

	Header	Filter Value	Partition	Company Prefix	Individual Asset Reference
GIAI-96	8	3	3	20-40	62-42
	0011 0100 (Binary value)	(Refer to Table 21 for values )	(Refer to Table 23 for values )	999,999 – 999,999,9 99,999 (Max. decimal range*)	4,611,686,018,427, 387,903 – 4,398,046,511,103 (Max. decimal range*)

1397

1398 \*Max. decimal value range of Company Prefix and Individual Asset Reference fields vary according to

1399 contents of the Partition field.

1400 **Table 22.** The EPC 96-bit GIAI bit allocation, header, and maximum decimal values.

- 1401 • *Header* is 8-bits, with a binary value of 0011 0100.
- 1402 • *Filter Value* is not part of the GIAI or EPC identifier, but is used for fast filtering and  
1403 pre-selection of basic asset types. The Filter Values for 64-bit and 96-bit GIAI are  
1404 the same. See Table 21.
- 1405 • The *Partition* is an indication of where the subsequent Company Prefix and  
1406 Individual Asset Reference numbers are divided. This organization matches the  
1407 structure in the EAN.UCC GIAI in which the Company Prefix may vary from 6 to 12  
1408 digits. The available values of *Partition* and the corresponding sizes of the *Company*  
1409 *Prefix* and *Asset Reference* fields are defined in Table 23.

Partition Value ( <i>P</i> )	Company Prefix		Individual Asset Reference	
	Bits ( <i>M</i> )	Digits ( <i>L</i> )	Bits ( <i>N</i> )	Digits
0	40	12	42	12
1	37	11	45	13
2	34	10	48	14
3	30	9	52	15
4	27	8	55	16
5	24	7	58	17
6	20	6	62	18

1410 **Table 23.** GIAI-96 Partitions.

- 1411 • *Company Prefix* contains a literal embedding of the Company Prefix.
- 1412 • *Individual Asset Reference* is a unique number for each instance. The EPC  
1413 representation is only capable of representing a subset of asset references allowed in  
1414 the General EAN.UCC Specifications. The capacity of this asset reference is less than  
1415 the maximum EAN.UCC System specification for asset references, no leading zeros  
1416 are permitted, and only numbers are permitted.

### 1417 **3.8.2.1 GIAI-96 Encoding Procedure**

1418 The following procedure creates a GIAI-96 encoding.

1419 Given:

1420 An EAN.UCC GIAI consisting of digits  $d_1d_2\dots d_K$ , where  $K \leq 30$ .

1421 The length  $L$  of the Company Prefix portion of the GIAI

1422 A Filter Value  $F$  where  $0 \leq F < 8$

1423 Procedure:

- 1424 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column  
 1425 of the Partition Table (Table 23) to determine the Partition Value,  $P$ , the number of bits  
 1426  $M$  in the Company Prefix field, and the number of bits  $N$  in the Individual Asset  
 1427 Reference field. If  $L$  is not found in any row of Table 23, stop: this GIAI cannot be  
 1428 encoded in a GIAI-96.
- 1429 2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_L$  and considering the  
 1430 result to be a decimal integer,  $C$ .
- 1431 3. Construct the Individual Asset Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_K$ . If  
 1432 any of these characters is not a digit, stop: this GIAI cannot be encoded in the GIAI-96  
 1433 encoding. Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^N$ , stop: this  
 1434 GIAI cannot be encoded in the GIAI-96 encoding. Also, if  $K > L+1$  and  $d_{(L+1)} = 0$ , stop:  
 1435 this GIAI cannot be encoded in the GIAI-96 encoding (because leading zeros are not  
 1436 permitted except in the case where the Individual Asset Reference consists of a single  
 1437 zero digit).
- 1438 4. Construct the final encoding by concatenating the following bit fields, from most  
 1439 significant to least significant: Header 00110100 (8 bits), Filter Value  $F$  (3 bits),  
 1440 Partition Value  $P$  from Step 2 (3 bits), Company Prefix  $C$  from Step 3 ( $M$  bits),  
 1441 Individual Asset Number  $S$  from Step 4 ( $N$  bits). Note that  $M+N = 82$  bits for all  $P$ .

### 1442 3.8.2.2 GIAI-96 Decoding Procedure

1443 Given:

1444 A GIAI-96 as a 96-bit bit string  $00110100b_{87}b_{86}\dots b_0$  (where the first eight bits  
 1445  $00110100$  are the header)

1446 Yields:

1447 An EAN.UCC GIAI

1448 A Filter Value

1449 Procedure:

- 1450 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1451 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
 1452  $P = 7$ , stop: this bit string cannot be decoded as a GIAI-96.
- 1453 3. Look up the Partition Value  $P$  in Table 23 to obtain the number of bits  $M$  in the  
 1454 Company Prefix and the number of digits  $L$  in the Company Prefix.
- 1455 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned  
 1456 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a  
 1457 legal GIAI-96 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ ,  
 1458 adding leading zeros as necessary to make up  $L$  digits in total.
- 1459 5. Extract the Individual Asset Reference by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_0$  as an  
 1460 unsigned integer. If this integer is greater than or equal to  $10^{(30-L)}$ , stop: the input bit  
 1461 string is not a legal GIAI-96 encoding. Otherwise, convert this integer to a decimal

1462 number  $s_1s_2\dots s_J$ , with no leading zeros (exception: if the integer is equal to zero, convert  
 1463 it to a single zero digit).

1464 6. Construct a K-digit number  $d_1d_2\dots d_K$  where  $d_1d_2\dots d_L = p_1p_2\dots p_L$  from Step 4, and  
 1465  $d_{(L+1)}d_{(L+2)}\dots d_K = s_1s_2\dots s_J$  from Step 5. This K-digit number, where  $K \leq 30$ , is the  
 1466 EAN.UCC GIAI.

1467 **3.9 DoD Tag Data Constructs (non-normative)**

1468 **3.9.1 DoD-64**

1469 This tag data construct may be used to encode 64-bit Class 0 and Class 1 tags for  
 1470 shipping goods to the United States Department of Defense by a supplier who has already  
 1471 been assigned a CAGE (Commercial and Government Entity) code.

1472 At the time of this writing, the details of what information to encode into these fields is  
 1473 explained in a document titled "United States Department of Defense Supplier's Passive  
 1474 RFID Information Guide" that can be obtained at the United States Department of  
 1475 Defense's web site (<http://www.dodrfid.org/supplierguide.htm>).

1476 Currently, the basic encoding structure of DOD-64 Tag Data Construct is as below.

1477

	Header	Filter Value	Government Managed Identifier	Serial Number
DoD-64	8	2	30	24
	1100 1110 (Binary value)	(Consult proper US Dept. Defense document for details)	Encoded with supplier CAGE code in truncated ASCII format (Consult proper US Dept. Defense document for details)	16,777,215 (Max. decimal value)

1478 **Table 24.** The DoD-64 bit allocation, header, and maximum decimal values

1479 **3.9.2 DoD-96**

1480 This tag data construct may be used to encode 96-bit Class 0 and Class 1 tags for  
 1481 shipping goods to the United States Department of Defense by a supplier who has already  
 1482 been assigned a CAGE (Commercial and Government Entity) code.

1483 At the time of this writing, the details of what information to encode into these fields is  
 1484 explained in a document titled "United States Department of Defense Supplier's Passive  
 1485 RFID Information Guide" that can be obtained at the United States Department of  
 1486 Defense's web site (<http://www.dodrfid.org/supplierguide.htm>).

1487 Currently, the basic encoding structure of DoD-96 Tag Data Construct is as below.

	Header	Filter Value	Government Managed Identifier	Serial Number
DoD-96	8	4	48	36
	0010 1111 (Binary value)	(Consult proper US Dept. Defense document for details)	Encoded with supplier CAGE code in 8-bit ASCII format (Consult US Dept. Defense doc for details)	68,719,476,735 (Max. decimal value)

1488 **Table 25.** The DoD-96 bit allocation, header, and maximum decimal values

1489

## 1490 **4 URI Representation**

1491 This section defines standards for the encoding of the Electronic Product Code™ as a  
 1492 Uniform Resource Identifier (URI). The URI Encoding complements the EPC Tag  
 1493 Encodings defined for use within RFID tags and other low-level architectural  
 1494 components. URIs provide a means for application software to manipulate Electronic  
 1495 Product Codes in a way that is independent of any particular tag-level representation,  
 1496 decoupling application logic from the way in which a particular Electronic Product Code  
 1497 was obtained from a tag.

1498 This section defines four categories of URI. The first are URIs for pure identities,  
 1499 sometimes called “canonical forms.” These contain only the unique information that  
 1500 identifies a specific physical object, and are independent of tag encodings. The second  
 1501 category are URIs that represent specific tag encodings. These are used in software  
 1502 applications where the encoding scheme is relevant, as when commanding software to  
 1503 write a tag. The third category are URIs that represent patterns, or sets of EPCs. These  
 1504 are used when instructing software how to filter tag data. The last category is a URI  
 1505 representation for raw tag information, generally used only for error reporting purposes.

1506 All categories of URIs are represented as Uniform Reference Names (URNs) as defined  
 1507 by [RFC2141], where the URN Namespace is `epc`.

1508 This section complements Section 3, EPC Bit-level Encodings, which specifies the  
 1509 currently defined tag-level representations of the Electronic Product Code.

### 1510 **4.1 URI Forms for Pure Identities**

1511 (This section is non-normative; the formal specifications for the URI types are given in  
 1512 Sections 4.3 and 5.)

1513 URI forms are provided for pure identities, which contain just the EPC fields that serve to  
1514 distinguish one object from another. These URIs take the form of Universal Resource  
1515 Names (URNs), with a different URN namespace allocated for each pure identity type.

1516 For the EPC General Identifier (Section 2.1.1), the pure identity URI representation is as  
1517 follows:

1518 `urn:epc:id:gid:GeneralManagerNumber.ObjectClass.SerialNumber`

1519 In this representation, the three fields *GeneralManagerNumber*, *ObjectClass*,  
1520 and *SerialNumber* correspond to the three components of an EPC General Identifier  
1521 as described in Section 2.1.1. In the URI representation, each field is expressed as a  
1522 decimal integer, with no leading zeros (except where a field's value is equal to zero, in  
1523 which case a single zero digit is used).

1524 There are also pure identity URI forms defined for identity types corresponding to certain  
1525 types within the EAN.UCC System family of codes as defined in Section 2.1.2; namely,  
1526 the Serialized Global Trade Item Number (SGTIN), the Serial Shipping Container Code  
1527 (SSCC), the Serialized Global Location Number (SGLN), the Global Reusable Asset  
1528 Identifier (GRAI), and the Global Individual Asset Identifier (GIAI). The URI  
1529 representations corresponding to these identifiers are as follows:

1530 `urn:epc:id:sgtin:CompanyPrefix.ItemReference.SerialNumber`

1531 `urn:epc:id:sscc:CompanyPrefix.SerialReference`

1532 `urn:epc:id:sgln:CompanyPrefix.LocationReference.SerialNumber`

1533 `urn:epc:id:grai:CompanyPrefix.AssetType.SerialNumber`

1534 `urn:epc:id:giai:CompanyPrefix.IndividualAssetReference`

1535 In these representations, *CompanyPrefix* corresponds to an EAN.UCC company  
1536 prefix assigned to a manufacturer by the UCC or EAN. (A UCC company prefix is  
1537 converted to an EAN.UCC company prefix by adding one leading zero at the beginning.)  
1538 The number of digits in this field is significant, and leading zeros are included as  
1539 necessary.

1540 The *ItemReference*, *SerialReference*, *LocationReference*, and  
1541 *AssetType* fields correspond to the similar fields of the GTIN, SSCC, GLN, and GRAI,  
1542 respectively. Like the *CompanyPrefix* field, the number of digits in these fields is  
1543 significant, and leading zeros are included as necessary. The number of digits in these  
1544 fields, when added to the number of digits in the *CompanyPrefix* field, always total  
1545 the same number of digits according to the identity type: 13 digits total for SGTIN, 17  
1546 digits total for SSCC, 12 digits total for SGLN, and 12 characters total for the GRAI.  
1547 (The *ItemReference* field of the SGTIN includes the GTIN Indicator (PI) digit,  
1548 appended to the beginning of the item reference. The *SerialReference* field  
1549 includes the SSCC Extension Digit (ED), appended at the beginning of the serial  
1550 reference. In no case are check digits included in URI representations.)

1551 In contrast to the other fields, the *SerialNumber* field of the SGLN is a pure integer,  
1552 with no leading zeros. The *SerialNumber* field of the SGTIN and GRAI, as well as  
1553 the *IndividualAssetReference* field of the GIAI, may include digits, letters, and

1554 certain other characters. In order for an SGTIN, GRAI, or GIAI to be encodable on a 64-  
1555 bit and 96-bit tag, however, these fields must consist only of digits with no leading zeros.  
1556 These restrictions are defined in the encoding procedures for these types, as well as in  
1557 Appendix F.

1558 An SGTIN, SSCC, etc in this form is said to be in SGTIN-URI form, SSCC-URI form,  
1559 etc form, respectively. Here are examples:

1560 `urn:epc:id:sgtin:0652642.800031.400`

1561 `urn:epc:id:sscc:0652642.0123456789`

1562 `urn:epc:id:sgln:0652642.12345.400`

1563 `urn:epc:id:grai:0652642.12345.1234`

1564 `urn:epc:id:giai:0652642.123456`

1565 Referring to the first example, the corresponding GTIN-14 code is 80652642000311.  
1566 This divides as follows: the first digit (8) is the PI digit, which appears as the first digit  
1567 of the *ItemReference* field in the URI, the next seven digits (0652642) are the  
1568 *CompanyPrefix*, the next five digits (00031) are the remainder of the  
1569 *ItemReference*, and the last digit (1) is the check digit, which is not included in the  
1570 URI.

1571 Referring to the second example, the corresponding SSCC is 006526421234567896 and  
1572 the last digit (6) is the check digit, not included in the URI.

1573 Referring to the third example, the corresponding GLN is 0652642123458, where the last  
1574 digit (8) is the check digit, not included in the URI.

1575 Referring to the fourth example, the corresponding GRAI is 006526421234581234,  
1576 where the digit (8) is the check digit, not included in the URI.

1577 Referring to the fifth example, the corresponding GIAI is 0652642123456. (GIAI codes  
1578 do not include a check digit.)

1579 Note that all five URI forms have an explicit indication of the division between the  
1580 company prefix and the remainder of the code. This is necessary so that the URI  
1581 representation may be converted into tag encodings. In general, the URI representation  
1582 may be converted to the corresponding EAN.UCC numeric form (by combining digits  
1583 and calculating the check digit), but converting from the EAN.UCC numeric form to the  
1584 corresponding URI representation requires independent knowledge of the length of the  
1585 company prefix.

1586 For the DoD identifier (Section 3.9), the pure identity URI representation is as follows:

1587 `urn:epc:id:usdod:CAGECodeOrDODAAC.serialNumber`

1588 where *CAGECodeOrDODAAC* is the five-character CAGE code or six-character  
1589 DoDAAC, and *serialNumber* is the serial number represented as a decimal integer  
1590 with no leading zeros (except that a serial number whose value is zero should be  
1591 represented as a single zero digit). Note that a space character is never included as part of



1592 *CAGECodeOrDODAAC* in the URI form, even though on a 96-bit tag a space character is  
1593 used to pad the five-character CAGE code to fit into the six-character field on the tag.  
1594

## 1595 **4.2 URI Forms for Related Data Types**

1596 (This section is non-normative; the formal specifications for the URI types are given in  
1597 Sections 4.3 and 5.)

1598 There are several data types that commonly occur in applications that manipulate  
1599 Electronic Product Codes, which are not themselves Electronic Product Codes but are  
1600 closely related. This specification provides URI forms for those as well. The general  
1601 form of the *epc* URN Namespace is

1602 *urn:epc:type:typeSpecificPart*

1603 The *type* field identifies a particular data type, and *typeSpecificPart* encodes  
1604 information appropriate for that data type. Currently, there are three possibilities defined  
1605 for *type*, discussed in the next three sections.

### 1606 **4.2.1 URIs for EPC Tags**

1607 In some cases, it is desirable to encode in URI form a specific tag encoding of an EPC.  
1608 For example, an application may wish to report to an operator what kinds of tags have  
1609 been read. In another example, an application responsible for programming tags needs to  
1610 be told not only what Electronic Product Code to put on a tag, but also the encoding  
1611 scheme to be used. Finally, applications that wish to manipulate any additional data  
1612 fields on tags need some representation other than the pure identity forms.

1613 EPC Tag URIs are encoded by setting the *type* field to *tag*, with the entire URI having  
1614 this form:

1615 *urn:epc:tag:EncName:EncodingSpecificFields*

1616 where *EncName* is the name of an EPC encoding scheme, and  
1617 *EncodingSpecificFields* denotes the data fields required by that encoding  
1618 scheme, separated by dot characters. Exactly what fields are present depends on the  
1619 specific encoding scheme used.

1620 In general, there are one or more encoding schemes (and corresponding *EncName*  
1621 values) defined for each pure identity type. For example, the SGTIN Identifier has two  
1622 encodings defined: *sgtin-96* and *sgtin-64*, corresponding to the 96-bit encoding  
1623 and the 64-bit encoding. Note that these encoding scheme names are in one-to-one  
1624 correspondence with unique tag Header values, which are used to represent the encoding  
1625 schemes on the tag itself.

1626 The *EncodingSpecificFields*, in general, include all the fields of the  
1627 corresponding pure identity type, possibly with additional restrictions on numeric range,  
1628 plus additional fields supported by the encoding. For example, all of the defined  
1629 encodings for the Serialized GTIN include an additional Filter Value that applications use

1630 to do tag filtering based on object characteristics associated with (but not encoded within)  
1631 an object's pure identity.

1632 Here is an example: a Serialized GTIN 64-bit encoding:

1633 urn:epc:tag:sgtin-64:3.0652642.800031.400

1634 In this example, the number 3 is the Filter Value .

1635 The tag URI for the DoD identifier is as follows:

1636 urn:epc:tag:tagType:filter.CAGECodeOrDODAAC.serialNumber

1637 where *tagType* is either *usdod-64* or *usdod-96*, *filter* is the filter value represented  
1638 as either one or two decimal digits (depending on the *tagType*), and the other two fields  
1639 are as defined above in 4.1.

1640

#### 1641 4.2.2 URIs for Raw Bit Strings Arising From Invalid Tags

1642 Certain bit strings do not correspond to legal encodings. For example, if the most  
1643 significant bits cannot be recognized as a valid EPC header, the bit-level pattern is not a  
1644 legal EPC. For a second example, if the binary value of a field in a tag encoding is  
1645 greater than the value that can be contained in the number of decimal digits in that field  
1646 in the URI form, the bit level pattern is not a legal EPC. Nevertheless, software may wish  
1647 to report such invalid bit-level patterns to users or to other software, and so a  
1648 representation of invalid bit-level patterns as URIs is provided. The *raw* form of the URI  
1649 has this general form:

1650 urn:epc:raw:BitLength.Value

1651 where *BitLength* is the number of bits in the invalid representation, and *Value* is the  
1652 entire bit-level representation converted to a single hexadecimal number and preceded by  
1653 the letter "x". For example, this bit string:

1654 0000000000000000000000000000000010010001101001101111010101101101111011101111

1655 which is invalid because no valid header begins with 0000 0000, corresponds to this raw  
1656 URI:

1657 urn:epc:raw:64.x00001234DEADBEEF

1658 In order to ensure that a given bit string has only one possible raw URI representation,  
1659 the number of digits in the hexadecimal value is required to be equal to the *BitLength*  
1660 divided by four and rounded up to the nearest whole number. Moreover, only uppercase  
1661 letters are permitted for the hexadecimal digits A, B, C, D, E, and F.

1662 It is intended that this URI form be used only when reporting errors associated with  
1663 reading invalid tags. It is *not* intended to be a general mechanism for communicating  
1664 arbitrary bit strings for other purposes.

1665 *Explanation (non-normative): The reason for recommending against using the raw URI*  
1666 *for general purposes is to avoid having an alternative representation for legal tag*  
1667 *encodings.*

1668 Earlier versions of this specification described a decimal, as opposed to hexadecimal,  
1669 version of the raw URI. This is still supported for back-compatibility, but its use is no  
1670 longer recommended. The “x” character is included so that software may distinguish  
1671 between the decimal and hexadecimal forms.

### 1672 4.2.3 URIs for EPC Patterns

1673 Certain software applications need to specify rules for filtering lists of EPCs according to  
1674 various criteria. This specification provides a *pattern* URI form for this purpose. A  
1675 pattern URI does not represent a single Electronic Product Code, but rather refers to a set  
1676 of EPCs. A typical pattern looks like this:

1677 `urn:epc:pat:sgtin-64:3.0652642.[1024-2047].*`

1678 This pattern refers to any EPC SGTIN Identifier 64-bit tag, whose Filter field is 3, whose  
1679 Company Prefix is 0652642, whose Item Reference is in the range  $1024 \leq \textit{itemReference}$   
1680  $\leq 2047$ , and whose Serial Number may be anything at all.

1681 In general, there is a pattern form corresponding to each tag encoding form  
1682 (Section 4.2.1), whose syntax is essentially identical except that ranges or the star (\*)  
1683 character may be used in each field.

1684 For the SGTIN, SSCC, SGLN, GRAI and GIAI patterns, the pattern syntax slightly  
1685 restricts how wildcards and ranges may be combined. Only two possibilities are  
1686 permitted for the *CompanyPrefix* field. One, it may be a star (\*), in which case the  
1687 following field (*ItemReference*, *SerialReference*, or *LocationReference*)  
1688 must also be a star. Two, it may be a specific company prefix, in which case the  
1689 following field may be a number, a range, or a star. A range may not be specified for the  
1690 *CompanyPrefix*.

1691 *Explanation (non-normative): Because the company prefix is variable length, a range*  
1692 *may not be specified, as the range might span different lengths. Also, in the case of the*  
1693 *SGTIN-64, SSCC-64, and GLN-64 encodings, the tag contains a manager index which*  
1694 *maps into a company prefix but not in a way that preserves contiguous ranges. When a*  
1695 *particular company prefix is specified, however, it is possible to match ranges or all*  
1696 *values of the following field, because its length is fixed for a given company prefix. The*  
1697 *other case that is allowed is when both fields are a star, which works for all tag*  
1698 *encodings because the corresponding tag fields (including the Partition field, where*  
1699 *present) are simply ignored.*

1700 The pattern URI for the DoD Construct is as follows:

1701 `urn:epc:pat:tagType:filterPat.CAGECodeOrDODAACPat.serialNumberPat`  
1702

1703 where *tagType* is as defined above in 4.2.1, *filterPat* is either a filter value, a  
1704 range of the form [ *lo-hi* ], or a \* character; *CAGECodeOrDODAACPat* is either a  
1705 CAGE Code/DODAAC or a \* character; and *serialNumberPat* is either a serial  
1706 number, a range of the form [ *lo-hi* ], or a \* character.

1707 **4.3 Syntax**

1708 The syntax of the EPC-URI and the URI forms for related data types are defined by the  
1709 following grammar.

1710 **4.3.1 Common Grammar Elements**

1711 NumericComponent ::= ZeroComponent | NonZeroComponent  
1712 ZeroComponent ::= "0"  
1713 NonZeroComponent ::= NonZeroDigit Digit\*  
1714 PaddedNumericComponent ::= Digit+  
1715 Digit ::= "0" | NonZeroDigit  
1716 NonZeroDigit ::= "1" | "2" | "3" | "4"  
1717 | "5" | "6" | "7" | "8" | "9"  
1718 UpperAlpha ::= "A" | "B" | "C" | "D" | "E" | "F" | "G"  
1719 | "H" | "I" | "J" | "K" | "L" | "M" | "N"  
1720 | "O" | "P" | "Q" | "R" | "S" | "T" | "U"  
1721 | "V" | "W" | "X" | "Y" | "Z"  
1722 LowerAlpha ::= "a" | "b" | "c" | "d" | "e" | "f" | "g"  
1723 | "h" | "i" | "j" | "k" | "l" | "m" | "n"  
1724 | "o" | "p" | "q" | "r" | "s" | "t" | "u"  
1725 | "v" | "w" | "x" | "y" | "z"  
1726 OtherChar ::= "!" | "'" | "(" | ")" | "\*" | "+" | "," | "-"  
1727 | "." | ":" | ";" | "=" | "\_"  
1728 UpperHexChar ::= Digit | "A" | "B" | "C" | "D" | "E" | "F"  
1729 HexComponent ::= UpperHexChar+  
1730 Escape ::= "%" HexChar HexChar  
1731 HexChar ::= Digit | "A" | "B" | "C" | "D" | "E" | "F"  
1732 | UpperHexChar | "a" | "b" | "c" | "d" | "e" | "f"  
1733 GS3A3Char ::= Digit | UpperAlpha | LowerAlpha | OtherChar  
1734 | Escape  
1735 GS3A3Component ::= GS3A3Char+

1736 The syntactic construct GS3A3Component is used to represent fields of EAN.UCC  
1737 codes that permit alphanumeric and other characters as specified in Figure 3A3-1 of the  
1738 EAN.UCC General Specifications. Owing to restrictions on URN syntax as defined by  
1739 [RFC2141], not all characters permitted in the EAN.UCC General Specifications may be  
1740 represented directly in a URN. Specifically, the characters " (double quote), % (percent),  
1741 & (ampersand), / (forward slash), < (less than), > (greater than), and ? (question mark)  
1742 are permitted in the General Specifications but may not be included directly in a URN.  
1743 To represent one of these characters in a URN, escape notation must be used in which the

1744 character is represented by a percent sign, followed by two hexadecimal digits that give  
1745 the ASCII character code for the character.

#### 1746 **4.3.2 EPCGID-URI**

1747 EPCGID-URI ::= "urn:epc:id:gid:" 2\*(NumericComponent ".")  
1748 NumericComponent

#### 1749 **4.3.3 SGTIN-URI**

1750 SGTIN-URI ::= "urn:epc:id:sgtin:" SGTINURIBody

1751 SGTINURIBody ::= 2\*(PaddedNumericComponent ".")  
1752 GS3A3Component

1753 The number of characters in the two PaddedNumericComponent fields must total 13  
1754 (not including any of the dot characters).

1755 The Serial Number field of the SGTIN-URI is expressed as a GS3A3Component,  
1756 which permits the representation of all characters permitted in the UCC/EAN-128  
1757 Application Identifier 21 Serial Number according to the EAN.UCC General  
1758 Specifications. SGTIN-URIs that are derived from 64-bit and 96-bit tag encodings,  
1759 however, will have Serial Numbers that consist only of digit characters and which have  
1760 no leading zeros. These limitations are described in the encoding procedures, and in  
1761 Appendix F.

#### 1762 **4.3.4 SSCC-URI**

1763 SSCC-URI ::= "urn:epc:id:sscc:" SSCCURIBody

1764 SSCCURIBody ::= PaddedNumericComponent "."  
1765 PaddedNumericComponent

1766 The number of characters in the two PaddedNumericComponent fields must total 17  
1767 (not including any of the dot characters).

#### 1768 **4.3.5 SGLN-URI**

1769 SGLN-URI ::= "urn:epc:id:sgln:" SGLNURIBody

1770 SGLNURIBody ::= 2\*(PaddedNumericComponent ".")  
1771 NumericComponent

1772 The number of characters in the two PaddedNumericComponent fields must total 12  
1773 (not including any of the dot characters).

#### 1774 **4.3.6 GRAI-URI**

1775 GRAI-URI ::= "urn:epc:id:grai:" GRAIURIBody

1776 GRAIURIBody ::= 2\*(PaddedNumericComponent ".")  
1777 GS3A3Component

1778 The number of characters in the two `PaddedNumericComponent` fields must total 12  
1779 (not including any of the dot characters).

1780 The Serial Number field of the GRAI-URI is expressed as a `GS3A3Component`, which  
1781 permits the representation of all characters permitted in the Serial Number field of the  
1782 GRAI according to the EAN.UCC General Specifications. GRAI-URIs that are derived  
1783 from 64-bit and 96-bit tag encodings, however, will have Serial Numbers that consist  
1784 only of digit characters and which have no leading zeros. These limitations are described  
1785 in the encoding procedures, and in Appendix F.

#### 1786 **4.3.7 GIAI-URI**

1787 `GIAI-URI ::= "urn:epc:id:giai:" GIAIURIBody`

1788 `GIAIURIBody ::= PaddedNumericComponent "." GS3A3Component`

1789 The total number of characters in the `PaddedNumericComponent` and  
1790 `GS3A3Component` fields must not exceed 30 (not including the dot character that  
1791 separates the two fields).

1792 The Individual Asset Reference field of the GIAI-URI is expressed as a  
1793 `GS3A3Component`, which permits the representation of all characters permitted in the  
1794 Individual Asset Reference field of the GIAI according to the EAN.UCC General  
1795 Specifications. GIAI-URIs that are derived from 64-bit and 96-bit tag encodings,  
1796 however, will have Individual Asset References that consist only of digit characters and  
1797 which have no leading zeros. These limitations are described in the encoding procedures,  
1798 and in Appendix F.

#### 1799 **4.3.8 EPC Tag URI**

1800 `TagURI ::= "urn:epc:tag:" TagURIBody`

1801 `TagURIBody ::= GIDTagURIBody | SGTINSGLNGRAITagURIBody |`  
1802 `SSCCGIAITagURIBody`

1803 `GIDTagURIBody ::= GIDTagEncName ":" 2*(NumericComponent ".")`  
1804 `NumericComponent`

1805 `GIDTagEncName ::= "gid-96"`

1806 `SGTINSGLNGRAITagURIBody ::= SGTINSGLNGRAITagEncName ":"`  
1807 `NumericComponent "." 2*(PaddedNumericComponent ".")`  
1808 `NumericComponent`

1809 `SGTINSGLNGRAITagEncName ::= "sgtin-96" | "sgtin-64" | "sgln-`  
1810 `96" | "sgln-64" | "grai-96" | "grai-64"`

1811 `SSCCGIAITagURIBody ::= SSCCGIAITagEncName ":"`  
1812 `NumericComponent 2*(". PaddedNumericComponent")`

1813 `SSCCGIAITagEncName ::= "sscc-96" | "sscc-64" | "giai-96" |`  
1814 `"giai-64"`

1815 **4.3.9 Raw Tag URI**

1816 RawURI ::= "urn:epc:raw:" RawURIBody( DecimalRawURIBody |  
1817 HexRawURIBody )  
1818 DecimalRawURIBody ::= NonZeroComponent "." NumericComponent  
1819 HexRawURIBody ::= NonZeroComponent ".x" HexComponent

1820 **4.3.10 EPC Pattern URI**

1821 PatURI ::= "urn:epc:pat:" PatBody  
1822 PatBody ::= GIDPatURIBody | SGTINSGLNGRAIPatURIBody |  
1823 SSCGIAIPatURIBody  
1824 GIDPatURIBody ::= GIDTagEncName ":" 2\*(PatComponent ".")  
1825 PatComponent  
1826 SGTINSGLNGRAIPatURIBody ::= SGTINSGLNGRAITagEncName ":"  
1827 PatComponent ". " GS1PatBody ". " PatComponent  
1828 SSCGIAIPatURIBody ::= SSCGIAITagEncName ":" PatComponent  
1829 ". " GS1PatBody  
1830 GS1PatBody ::= "\*.\*" | ( PaddedNumericComponent ". "  
1831 PatComponent )  
1832 PatComponent ::= NumericComponent  
1833 | StarComponent  
1834 | RangeComponent  
1835 StarComponent ::= "\*"   
1836 RangeComponent ::= "[ " NumericComponent "-"  
1837 NumericComponent "]"  
1838 For a RangeComponent to be legal, the numeric value of the first  
1839 NumericComponent must be less than or equal to the numeric value of the second  
1840 NumericComponent.

1841 **4.3.11 DoD Construct URI**

1842 DOD-URI ::= "urn:epc:id:usdod:" CAGECodeOrDODAAC ". "  
1843 DoDSerialNumber  
1844 DODTagURI ::= "urn:epc:tag:" DoDTagType ":" DoDFilter ". "  
1845 CAGECodeOrDODAAC ". " DoDSerialNumber  
1846 DODPatURI ::= "urn:epc:pat:" DoDTagType ":" DoDFilterPat ". "  
1847 CAGECodeOrDODAACPat ". " DoDSerialNumberPat  
1848 DoDTagType ::= "usdod-64" | "usdod-96"  
1849 DoDFilter ::= NumericComponent  
1850 CAGECodeOrDODAAC ::= CAGECode | DODAAC

```

1851 CAGECode ::= CAGECodeOrDODAACChar*5
1852 DODAAC ::= CAGECodeOrDODAACChar*6
1853 DoDSerialNumber ::= NumericComponent
1854 DoDFilterPat ::= PatComponent
1855 CAGECodeOrDODAACPat ::= CAGECodeOrDODAAC | StarComponent
1856 DoDSerialNumberPat ::= PatComponent
1857 CAGECodeOrDODAACChar ::= Digit | "A" | "B" | "C" | "D" | "E"
1858 | "F" | "G" | "H" | "J" | "K" | "L" | "M" | "N" | "P" | "Q"
1859 | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
1860

```

### 1861 **4.3.12 Summary (non-normative)**

1862 The syntax rules above can be summarized informally as follows:

```

1863 urn:epc:id:gid:MMM.CCC.SSS
1864 urn:epc:id:sgtin:PPP.III.SSS
1865 urn:epc:id:sscc:PPP.III
1866 urn:epc:id:sgln:PPP.III
1867 urn:epc:id:grai:PPP.III.SSS
1868 urn:epc:id:giai:PPP.SSS
1869 urn:epc:id:usdod:TTT.SSS
1870
1871 urn:epc:tag:sgtin-64:FFF.PPP.III.SSS
1872 urn:epc:tag:sscc-64:FFF.PPP.III
1873 urn:epc:tag:sgln-64:FFF.PPP.III.SSS
1874 urn:epc:tag:grai-64:FFF.PPP.III.SSS
1875 urn:epc:tag:giai-64:FFF.PPP.SSS
1876 urn:epc:tag:gid-96:MMM.CCC.SSS
1877 urn:epc:tag:sgtin-96:FFF.PPP.III.SSS
1878 urn:epc:tag:sscc-96:FFF.PPP.III
1879 urn:epc:tag:sgln-96:FFF.PPP.III.SSS
1880 urn:epc:tag:grai-96:FFF.PPP.III.SSS
1881 urn:epc:tag:giai-96:FFF.PPP.SSS
1882 urn:epc:tag:usdod-64:FFF.TTT.SSS
1883 urn:epc:tag:usdod-96:FFF.TTT.SSS

```



1884  
1885 urn:epc:raw:LLL.BBB  
1886 urn:epc:raw:LLL.HHH  
1887  
1888 urn:epc:pat:sgtin-64:FFFpat.PPP.IIIpat.SSSpat  
1889 urn:epc:pat:sgtin-64:FFFpat.\*.\*.SSSpat  
1890 urn:epc:pat:sscc-64:FFFpat.PPP.IIIpat  
1891 urn:epc:pat:sscc-64:FFFpat.\*.\*  
1892 urn:epc:pat:sgln-64:FFFpat.PPP.IIIpat.SSSpat  
1893 urn:epc:pat:sgln-64:FFFpat.\*.\*.SSSpat  
1894 urn:epc:pat:grai-64:FFFpat.PPP.IIIpat.SSSpat  
1895 urn:epc:pat:grai-64:FFFpat.\*.\*.SSSpat  
1896 urn:epc:pat:giai-64:FFFpat.PPP.SSSpat  
1897 urn:epc:pat:giai-64:FFFpat.\*.\*  
1898 urn:epc:pat:usdod-64:FFFpat.TTT.SSSpat  
1899 urn:epc:pat:usdod-64:FFFpat.\*.\*.SSSpat  
1900 urn:epc:pat:gid-96:MMMpat.CCCpat.SSSpat  
1901 urn:epc:pat:sgtin-96:FFFpat.PPP.IIIpat.SSSpat  
1902 urn:epc:pat:sgtin-96:FFFpat.\*.\*.SSSpat  
1903 urn:epc:pat:sscc-96:FFFpat.PPP.IIIpat  
1904 urn:epc:pat:sscc-96:FFFpat.\*.\*  
1905 urn:epc:pat:sgln-96:FFFpat.PPP.IIIpat.SSSpat  
1906 urn:epc:pat:sgln-96:FFFpat.\*.\*.SSSpat  
1907 urn:epc:pat:grai-96:FFFpat.PPP.IIIpat.SSSpat  
1908 urn:epc:pat:grai-96:FFFpat.\*.\*.SSSpat  
1909 urn:epc:pat:giai-96:FFFpat.PPP.SSSpat  
1910 urn:epc:pat:giai-96:FFFpat.\*.\*  
1911 urn:epc:pat:usdod-96:FFFpat.TTT.SSSpat  
1912 urn:epc:pat:usdod-96:FFFpat.\*.\*.SSSpat  
1913 where  
1914 MMM denotes a General Manager Number  
1915 CCC denotes an Object Class number

- 1916 *SSS* denotes a Serial Number or GIAI Individual Asset Reference
- 1917 *PPP* denotes an EAN.UCC Company Prefix
- 1918 *TTT* denotes a US DoD assigned CAGE code or DODAAC
- 1919 *III* denotes an SGTIN Item Reference (with Indicator Digit appended to the  
 1920 beginning), an SSCC Shipping Container Serial Number (with the Extension (ED) digit  
 1921 appended at the beginning), a SGLN Location Reference, or a GRAI Asset Type.
- 1922 *FFF* denotes a filter code as used by the SGTIN, SSCC, SGLN, GRAI, GIAI, and DoD  
 1923 tag encodings
- 1924 *XXXpat* is the same as *XXX* but allowing \* and [ l o - h i ] pattern syntax in addition
- 1925 *LLL* denotes the number of bits of an uninterpreted bit sequence
- 1926 *BBB* denotes the literal value of an uninterpreted bit sequence converted to decimal
- 1927 *HHH* denotes the literal value of an uninterpreted bit sequence converted to hexadecimal  
 1928 and preceded by the character 'x'.
- 1929 and where all numeric fields are in decimal with no leading zeros (unless the overall  
 1930 value of the field is zero, in which case it is represented with a single 0 character), with  
 1931 the exception of the hexadecimal raw representation.
- 1932 Exceptions:
- 1933 1. The length of *PPP* and *III* is significant, and leading zeros are used as necessary.  
 1934 The length of *PPP* is the length of the company prefix as assigned by EAN or  
 1935 UCC. The length of *III* plus the length of *PPP* must equal 13 for SGTIN, 17 for  
 1936 SSCC, 12 for GLN, or 12 for GRAI.
  - 1937 2. The Value field of `urn:epc:raw` is expressed in hexadecimal if the value is  
 1938 preceded by the character 'x'.

## 1939 **5 Translation between EPC-URI and Other EPC** 1940 **Representations**

1941 This section defines the semantics of EPC-URI encodings, by defining how they are  
 1942 translated into other EPC encodings and vice versa.

1943 The following procedure translates a bit-level encoding of an EPC into an EPC-URI:

- 1944 1. Determine the identity type and encoding scheme by finding the row in Table 1  
 1945 (Section 3.1) that matches the most significant bits of the bit string. If the most  
 1946 significant bits do not match any row of the table, stop: the bit string is invalid  
 1947 and cannot be translated into an EPC-URI. If the encoding scheme indicates one  
 1948 of the DoD Tag Data Constructs, consult the appropriate U.S. Department of  
 1949 Defense document for specific encoding and decoding rules. Otherwise, if the  
 1950 encoding scheme is SGTIN-64 or SGTIN-96, proceed to Step 2; if the encoding  
 1951 scheme is SSCC-64 or SSCC-96, proceed to Step 5; if the encoding scheme is  
 1952 SGLN-64 or SGLN-96, proceed to Step 8; if the encoding scheme is GRAI-64 or

- 1953 GRAI-96, proceed to Step 11; if the encoding scheme is GIAI-64 or GIAI-96,  
1954 proceed to Step 14; if the encoding scheme is GID-96, proceed to Step 17.
- 1955 2. Follow the decoding procedure given in Section 3.4.1.2 (for SGTIN-64) or in  
1956 Section 3.4.2.2 (for SGTIN-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ ,  
1957 the decimal Item Reference and Indicator  $i_1i_2\dots i_{(13-L)}$ , and the Serial Number  $S$ . If  
1958 the decoding procedure fails, stop: the bit-level encoding cannot be translated into  
1959 an EPC-URI.
- 1960 3. Create an EPC-URI by concatenating the following: the string  
1961 `urn:epc:id:sgtin:`, the Company Prefix  $p_1p_2\dots p_L$  where each digit  
1962 (including any leading zeros) becomes the corresponding ASCII digit character, a  
1963 dot (.) character, the Item Reference and Indicator  $i_1i_2\dots i_{(13-L)}$  (handled similarly),  
1964 a dot (.) character, and the Serial Number  $S$  as a decimal integer. The portion  
1965 corresponding to the Serial Number must have no leading zeros, except where the  
1966 Serial Number is itself zero in which case the corresponding URI portion must  
1967 consist of a single zero character.
- 1968 4. Go to Step 19.
- 1969 5. Follow the decoding procedure given in Section 3.5.1.2 (for SSCC-64) or in  
1970 Section 3.5.2.2 (for SSCC-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ ,  
1971 and the decimal Serial Reference  $s_1s_2\dots s_{(17-L)}$ . If the decoding procedure fails,  
1972 stop: the bit-level encoding cannot be translated into an EPC-URI.
- 1973 6. Create an EPC-URI by concatenating the following: the string  
1974 `urn:epc:id:sscc:`, the Company Prefix  $p_1p_2\dots p_L$  where each digit (including  
1975 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
1976 character, and the Serial Reference  $s_1s_2\dots s_{(17-L)}$  (handled similarly).
- 1977 7. Go to Step 19.
- 1978 8. Follow the decoding procedure given in Section 3.6.1.2 (for SGLN-64) or in  
1979 Section 3.6.2.2 (for SGLN-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ ,  
1980 the decimal Location Reference  $i_1i_2\dots i_{(12-L)}$ , and the Serial Number  $S$ . If the  
1981 decoding procedure fails, stop: the bit-level encoding cannot be translated into an  
1982 EPC-URI.
- 1983 9. Create an EPC-URI by concatenating the following: the string  
1984 `urn:epc:id:sgln:`, the Company Prefix  $p_1p_2\dots p_L$  where each digit (including  
1985 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
1986 character, the Location Reference  $i_1i_2\dots i_{(12-L)}$  (handled similarly), a dot (.)  
1987 character, and the Serial Number  $S$  as a decimal integer. The portion  
1988 corresponding to the Serial Number must have no leading zeros, except where the  
1989 Serial Number is itself zero in which case the corresponding URI portion must  
1990 consist of a single zero character.
- 1991 10. Go to Step 19.
- 1992 11. Follow the decoding procedure given in Section 3.7.1.2 (for GRAI-64) or in  
1993 Section 3.7.2.2 (for GRAI-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ , the

- 1994 decimal Asset Type  $i_1i_2\dots i_{(12-L)}$ , and the Serial Number  $S$ . If the decoding  
1995 procedure fails, stop: the bit-level encoding cannot be translated into an EPC-URI.
- 1996 12. Create an EPC-URI by concatenating the following: the string  
1997 `urn:epc:id:grai:`, the Company Prefix  $p_1p_2\dots p_L$  where each digit (including  
1998 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
1999 character, the Asset Type  $i_1i_2\dots i_{(12-L)}$  (handled similarly), a dot (.) character, and  
2000 the Serial Number  $S$  as a decimal integer. The portion corresponding to the Serial  
2001 Number must have no leading zeros, except where the Serial Number is itself zero  
2002 in which case the corresponding URI portion must consist of a single zero  
2003 character.
- 2004 13. Go to Step 19.
- 2005 14. Follow the decoding procedure given in Section 3.8.1.2 (for GIAI-64) or in  
2006 Section 3.8.2.2 (for GIAI-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ , and  
2007 the Individual Asset Reference  $S$ . If the decoding procedure fails, stop: the bit-  
2008 level encoding cannot be translated into an EPC-URI.
- 2009 15. Create an EPC-URI by concatenating the following: the string  
2010 `urn:epc:id:giai:`, the Company Prefix  $p_1p_2\dots p_L$  where each digit (including  
2011 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2012 character, and the Individual Asset Reference  $S$  as a decimal integer. The portion  
2013 corresponding to the Individual Asset Reference must have no leading zeros,  
2014 except where the Individual Asset Reference is itself zero in which case the  
2015 corresponding URI portion must consist of a single zero character.
- 2016 16. Go to Step 19.
- 2017 17. Follow the decoding procedure given in Section 3.3.1.2 to obtain the General  
2018 Manager Number  $M$ , the Object Class  $C$ , and the Serial Number  $S$ .
- 2019 18. Create an EPC-URI by concatenating the following: the string  
2020 `urn:epc:id:gid:`, the General Manager Number as a decimal integer, a dot  
2021 (.) character, the Object Class as a decimal integer, a dot (.) character, and the  
2022 Serial Number  $S$  as a decimal integer. Each decimal number must have no  
2023 leading zeros, except where the integer is itself zero in which case the  
2024 corresponding URI portion must consist of a single zero character.
- 2025 19. The translation is now complete.
- 2026 The following procedure translates a bit-level tag encoding into either an EPC Tag URI  
2027 or a Raw Tag URI:
- 2028 1. Determine the identity type and encoding scheme by finding the row in Table 1  
2029 (Section 3.1) that matches the most significant bits of the bit string. If the  
2030 encoding scheme indicates one of the DoD Tag Data Constructs, consult the  
2031 appropriate U.S. Department of Defense document for specific encoding and  
2032 decoding rules. If the encoding scheme is SGTIN-64 or SGTIN-96, proceed to  
2033 Step 2; if the encoding scheme is SSCC-64 or SSCC-96, proceed to Step 5; if the  
2034 encoding scheme is SGLN-64 or SGLN-96, proceed to Step 8; if the encoding

- 2035 scheme is GRAI-64 or GRAI-96, proceed to Step 11, if the encoding scheme is  
 2036 GIAI-64 or GIAI-96, proceed to Step 14, if the encoding scheme is GID-96,  
 2037 proceed to Step 17; otherwise, proceed to Step 20.
- 2038 2. Follow the decoding procedure given in Section 3.4.1.2 (for SGTIN-64) or in  
 2039 Section 3.4.2.2 (for SGTIN-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ ,  
 2040 the decimal Item Reference and Indicator  $i_1i_2\dots i_{(13-L)}$ , the Filter Value  $F$ , and the  
 2041 Serial Number  $S$ . If the decoding procedure fails, proceed to Step 20, otherwise  
 2042 proceed to the next step.
- 2043 3. Create an EPC Tag URI by concatenating the following: the string  
 2044 `urn:epc:tag:`, the encoding scheme (`sgtin-64` or `sgtin-96`), a colon (`:`)  
 2045 character, the Filter Value  $F$  as a decimal integer, a dot (`.`) character, the  
 2046 Company Prefix  $p_1p_2\dots p_L$  where each digit (including any leading zeros) becomes  
 2047 the corresponding ASCII digit character, a dot (`.`) character, the Item Reference  
 2048 and Indicator  $i_1i_2\dots i_{(13-L)}$  (handled similarly), a dot (`.`) character, and the Serial  
 2049 Number  $S$  as a decimal integer. The portions corresponding to the Filter Value  
 2050 and Serial Number must have no leading zeros, except where the corresponding  
 2051 integer is itself zero in which case a single zero character is used.
- 2052 4. Go to Step 21.
- 2053 5. Follow the decoding procedure given in Section 3.5.1.2 (for SSCC-64) or in  
 2054 Section 3.5.2.2 (for SSCC-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ ,  
 2055 and the decimal Serial Reference  $i_1i_2\dots s_{(17-L)}$ , and the Filter Value  $F$ . If the  
 2056 decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2057 6. Create an EPC Tag URI by concatenating the following: the string  
 2058 `urn:epc:tag:`, the encoding scheme (`sscc-64` or `sscc-96`), a colon (`:`)  
 2059 character, the Filter Value  $F$  as a decimal integer, a dot (`.`) character, the  
 2060 Company Prefix  $p_1p_2\dots p_L$  where each digit (including any leading zeros) becomes  
 2061 the corresponding ASCII digit character, a dot (`.`) character, and the Serial  
 2062 Reference  $i_1i_2\dots i_{(17-L)}$  (handled similarly).
- 2063 7. Go to Step 21.
- 2064 8. Follow the decoding procedure given in Section 3.6.1.2 (for SGLN-64) or in  
 2065 Section 3.6.2.2 (for SGLN-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ ,  
 2066 the decimal Location Reference  $i_1i_2\dots i_{(12-L)}$ , the Filter Value  $F$ , and the Serial  
 2067 Number  $S$ . If the decoding procedure fails, proceed to Step 20, otherwise proceed  
 2068 to the next step.
- 2069 9. Create an EPC Tag URI by concatenating the following: the string  
 2070 `urn:epc:tag:`, the encoding scheme (`sgln-64` or `sgln-96`), a colon (`:`)  
 2071 character, the Filter Value  $F$  as a decimal integer, a dot (`.`) character, the  
 2072 Company Prefix  $p_1p_2\dots p_L$  where each digit (including any leading zeros) becomes  
 2073 the corresponding ASCII digit character, a dot (`.`) character, the Location  
 2074 Reference  $i_1i_2\dots i_{(12-L)}$  (handled similarly), a dot (`.`) character, and the Serial  
 2075 Number  $S$  as a decimal integer. The portions corresponding to the Filter Value

- 2076 and Serial Number must have no leading zeros, except where the corresponding  
2077 integer is itself zero in which case a single zero character is used.
- 2078 10. Go to Step 21.
- 2079 11. Follow the decoding procedure given in Section 3.7.1.2 (for GRAI-64) or in  
2080 Section 3.7.2.2 (for GRAI-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ , the  
2081 decimal Asset Type  $i_1i_2\dots i_{(12-L)}$ , the Filter Value  $F$ , and the Serial Number  
2082  $d_{15}d_2\dots d_K$ . If the decoding procedure fails, proceed to Step 20, otherwise proceed  
2083 to the next step.
- 2084 12. Create an EPC Tag URI by concatenating the following: the string  
2085 `urn:epc:tag:`, the encoding scheme (`grai-64` or `grai-96`), a colon (`:`)  
2086 character, the Filter Value  $F$  as a decimal integer, a dot (`.`) character, the  
2087 Company Prefix  $p_1p_2\dots p_L$  where each digit (including any leading zeros) becomes  
2088 the corresponding ASCII digit character, a dot (`.`) character, the Asset Type  
2089  $s_1s_2\dots s_{(12-L)}$  (handled similarly), a dot (`.`) character, and the Serial Number  
2090  $d_{15}d_2\dots d_K$  as a decimal integer. The portions corresponding to the Filter Value  
2091 and Serial Number must have no leading zeros, except where the corresponding  
2092 integer is itself zero in which case a single zero character is used.
- 2093 13. Got to Step 21.
- 2094 14. Follow the decoding procedure given in Section 3.8.1.2 (for GIAI-64) or in  
2095 Section 3.8.2.2 (for GIAI-96) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ , the  
2096 decimal Individual Asset Reference  $s_1s_2\dots s_J$ , and the Filter Value  $F$ . If the  
2097 decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2098 15. Create an EPC Tag URI by concatenating the following: the string  
2099 `urn:epc:tag:`, the encoding scheme (`giai-64` or `giai-96`), a colon (`:`)  
2100 character, the Filter Value  $F$  as a decimal integer, a dot (`.`) character, the Company  
2101 Prefix  $p_1p_2\dots p_L$  where each digit (including any leading zeros) becomes the  
2102 corresponding ASCII digit character, a dot (`.`) character, and the Individual Asset  
2103 Reference  $i_1i_2\dots i_J$  (handled similarly). The portion corresponding to the Filter  
2104 Value must have no leading zeros, except where the corresponding integer is itself  
2105 zero in which case a single zero character is used.
- 2106 16. Go to Step 21.
- 2107 17. Follow the decoding procedure given in Section 3.3.1.2 to obtain the EPC  
2108 Manager Number, the Object Class, and the Serial Number.
- 2109 18. Create an EPC Tag URI by concatenating the following: the string  
2110 `urn:epc:tag:gid-96:`, the General Manager Number as a decimal number,  
2111 a dot (`.`) character, the Object Class as a decimal number, a dot (`.`) character, and  
2112 the Serial Number as a decimal number. Each decimal number must have no  
2113 leading zeros, except where the integer is itself zero in which case the  
2114 corresponding URI portion must consist of a single zero character.
- 2115 19. Go to Step 21.

2116 20. This tag is not a recognized EPC encoding, therefore create an EPC Raw URI by  
2117 concatenating the following: the string `urn:epc:raw:`, the length of the bit  
2118 string, a dot (.) character, a lowercase `x` character, and the value of the bit string  
2119 considered as a single hexadecimal integer. Both the length and the value must  
2120 have no leading zeros, except if the value is itself zero in which case a single zero  
2121 character is used.. The value must have a number of characters equal to the  
2122 length divided by four and rounded up to the nearest whole number, and must  
2123 only use uppercase letters for the hexadecimal digits A, B, C, D, E, and F.

2124 21. The translation is now complete.

2125

2126 The following procedure translates a URI into a bit-level EPC:

2127 1. If the URI is an SGTIN-URI (`urn:epc:id:sgtin:`), an SSCC-URI  
2128 (`urn:epc:id:sscc:`), an SGLN-URI (`urn:epc:id:sgln:`), a GRAI-  
2129 URI (`urn:epc:id:grai:`), a GIAI-URI (`urn:epc:id:giai:`), a GID-  
2130 URI (`urn:epc:id:gid:`), a DOD-URI (`urn:epc:id:usdod:`) or an EPC  
2131 Pattern URI (`urn:epc:pat:`), the URI cannot be translated into a bit-level  
2132 EPC.

2133 2. If the URI is a Raw Tag URI (`urn:epc:raw:`), create the bit-level EPC by  
2134 converting the second component of the Raw Tag URI into a binary integer,  
2135 whose length is equal to the first component of the Raw Tag URI. If the value of  
2136 the second component is too large to fit into a binary integer of that size, the URI  
2137 cannot be translated into a bit-level EPC.

2138 3. If the URI is an EPC Tag URI or US DoD Tag URI  
2139 (`urn:epc:tag:encName:`), parse the URI using the grammar for TagURI as  
2140 given in Section 4.3.8 or for DODTagURI as given in Section 4.3.11.. If the URI  
2141 cannot be parsed using these grammars, stop: the URI is illegal and cannot be  
2142 translated into a bit-level EPC. If `encName` is `usdod-96` or `usdod-64`,  
2143 consult the appropriate U.S. Department of Defense document for specific  
2144 translation rules. Otherwise, if `encName` is `sgtin-96` or `sgtin-64` go to  
2145 Step 4, if `encName` is `sscc-96` or `sscc-64` go to Step 9, if `encName` is  
2146 `sgln-96` or `sgln-64` go to Step 13, if `encName` is `grai-96` or `grai-64` go  
2147 to Step 18, if `encName` is `giai-96` or `giai-64` go to Step 22, or if `encName`  
2148 is `gid-96` go to Step 26.

2149 4. Let the URI be written as  
2150  $urn:epc:tag:encName:f_1f_2...f_F.p_1p_2...p_L.i_1i_2...i_{(13-L)}.s_1s_2...s_S.$

2151 5. Interpret  $f_1f_2...f_F$  as a decimal integer  $F$ .

2152 6. Interpret  $s_1s_2...s_S$  as a decimal integer  $S$ .

2153 7. Carry out the encoding procedure defined in Section 3.4.1.1 (SGTIN-64) or  
2154 Section 3.4.2.1 (SGTIN-96), using  $i_1p_1p_2...p_Li_2...i_{(13-L)}0$  as the EAN.UCC  
2155 GTIN-14 (the trailing zero is a dummy check digit, which is ignored by the

2156 encoding procedure),  $L$  as the length of the EAN.UCC company prefix,  $F$  from  
2157 Step 5 as the Filter Value, and  $S$  from Step 6 as the Serial Number. If the  
2158 encoding procedure fails because an input is out of range, or because the  
2159 procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.

2160 8. Go to Step 31.

2161 9. Let the URI be written as  
2162  $urn:epc:tag:encName:f_1f_2...f_F.p_1p_2...p_L.i_1i_2...i_{(17-L)}$ .

2163 10. Interpret  $f_1f_2...f_F$  as a decimal integer  $F$ .

2164 11. Carry out the encoding procedure defined in Section 3.5.1.1 (SSCC-64) or  
2165 Section 3.5.2.1 (SSCC-96), using  $i_1p_1p_2...p_Li_2i_3...i_{(17-L)}0$  as the EAN.UCC  
2166 SSCC,  $L$  as the length of the EAN.UCC company prefix, and  $F$  from Step 10 as  
2167 the Filter Value. If the encoding procedure fails because an input is out of range,  
2168 or because the procedure indicates a failure, stop: this URI cannot be encoded  
2169 into an EPC tag.

2170 12. Go to Step 31.

2171 13. Let the URI be written as  
2172  $urn:epc:tag:encName:f_1f_2...f_F.p_1p_2...p_L.i_1i_2...i_{(12-L)}.s_1s_2...s_S$ .

2173 14. Interpret  $f_1f_2...f_F$  as a decimal integer  $F$ .

2174 15. Interpret  $s_1s_2...s_S$  as a decimal integer  $S$ .

2175 16. Carry out the encoding procedure defined in Section 3.6.1.1 (SGLN-64) or  
2176 Section 3.6.2.1 (SGLN-96), using  $p_1p_2...p_Li_1i_2...i_{(12-L)}0$  as the EAN.UCC  
2177 GLN (the trailing zero is a dummy check digit, which is ignored by the encoding  
2178 procedure),  $L$  as the length of the EAN.UCC company prefix,  $F$  from Step 14 as  
2179 the Filter Value, and  $S$  from Step 15 as the Serial Number. If the encoding  
2180 procedure fails because an input is out of range, or because the procedure  
2181 indicates a failure, stop: this URI cannot be encoded into an EPC tag.

2182 17. Go to Step 31.

2183 18. Let the URI be written as  
2184  $urn:epc:tag:encName:f_1f_2...f_F.p_1p_2...p_L.i_1i_2...i_{(12-L)}.s_1s_2...s_S$ .

2185 19. Interpret  $f_1f_2...f_F$  as a decimal integer  $F$ .

2186 20. Carry out the encoding procedure defined in Section 3.7.1.1 (GRAI-64) or  
2187 Section 3.7.2.1 (GRAI-96), using  $0p_1p_2...p_Li_1i_2...i_{(12-L)}0s_1s_2...s_S$  as the  
2188 EAN.UCC GRAI (the second zero is a dummy check digit, which is ignored by  
2189 the encoding procedure),  $L$  as the length of the EAN.UCC company prefix, and  $F$   
2190 from Step 19 as the Filter Value. If the encoding procedure fails because an input  
2191 is out of range, or because the procedure indicates a failure, stop: this URI cannot  
2192 be encoded into an EPC tag.

2193 21. Go to Step 31.



- 2194 22. Let the URI be written as  
 2195  $\text{urn:epc:tag:encName:f}_1\text{f}_2\dots\text{f}_F.p_1p_2\dots p_L.s_1s_2\dots s_S$ .
- 2196 23. Interpret  $f_1f_2\dots f_F$  as a decimal integer  $F$ .
- 2197 24. Carry out the encoding procedure defined in Section 3.8.1.1 (GIAI-64) or  
 2198 Section 3.8.2.1 (GIAI-96), using  $p_1p_2\dots p_Ls_1s_2\dots s_S$  as the EAN.UCC GIAI,  $L$  as  
 2199 the length of the EAN.UCC company prefix, and  $F$  from Step 23 as the Filter  
 2200 Value. If the encoding procedure fails because an input is out of range, or  
 2201 because the procedure indicates a failure, stop: this URI cannot be encoded into  
 2202 an EPC tag.
- 2203 25. Go to Step 31.
- 2204 26. Let the URI be written as  
 2205  $\text{urn:epc:tag:encName:m}_1m_2\dots m_L.c_1c_2\dots c_K.s_1s_2\dots s_S$ .
- 2206 27. Interpret  $m_1m_2\dots m_L$  as a decimal integer  $M$ .
- 2207 28. Interpret  $c_1c_2\dots c_K$  as a decimal integer  $C$ .
- 2208 29. Interpret  $s_1s_2\dots s_S$  as a decimal integer  $S$ .
- 2209 30. Carry out the encoding procedure defined in Section 3.3.1.1 using  $M$  from Step 27  
 2210 as the General Manager Number,  $C$  from Step 28 as the Object Class, and  $S$  from  
 2211 Step 29 as the Serial Number. If the encoding procedure fails because an input is  
 2212 out of range, or because the procedure indicates a failure, stop: this URI cannot  
 2213 be encoded into an EPC tag.
- 2214 31. The translation is complete.

## 2215 6 Semantics of EPC Pattern URIs

2216 The meaning of an EPC Pattern URI ( $\text{urn:epc:pat:}$ ) can be formally defined as  
 2217 denoting a set of encoding-specific EPCs. The set of EPCs denoted by a specific EPC  
 2218 Pattern URI is defined by the following decision procedure, which says whether a given  
 2219 EPC Tag URI belongs to the set denoted by the EPC Pattern URI.

2220 Let  $\text{urn:epc:pat:EncName:P}_1.P_2\dots P_n$  be an EPC Pattern URI. Let  
 2221  $\text{urn:epc:tag:EncName:C}_1.C_2\dots C_n$  be an EPC Tag URI, where the *EncName*  
 2222 field of both URIs is the same. The number of components ( $n$ ) depends on the value of  
 2223 *EncName*.

2224 First, any EPC Tag URI component  $C_i$  is said to *match* the corresponding EPC Pattern  
 2225 URI component  $P_i$  if:

- 2226 •  $P_i$  is a `NumericComponent`, and  $C_i$  is equal to  $P_i$ ; or
- 2227 •  $P_i$  is a `PaddedNumericComponent`, and  $C_i$  is equal to  $P_i$  both in numeric value  
 2228 as well as in length; or
- 2229 •  $P_i$  is a `CAGECodeOrDODAAC`, and  $C_i$  is equal to  $P_i$ ; or

- 2230 •  $P_i$  is a RangeComponent [  $lo-hi$  ], and  $lo \leq C_i \leq hi$ ; or  
2231 •  $P_i$  is a StarComponent (and  $C_i$  is anything at all)  
2232 Then the EPC Tag URI is a member of the set denoted by the EPC Pattern URI if and  
2233 only if  $C_i$  matches  $P_i$  for all  $1 \leq i \leq n$ .

## 2234 **7 Background Information**

2237 This document draws from the previous work at the Auto-ID Center, and we recognize  
2238 the contribution of the following individuals: David Brock (MIT), Joe Foley (MIT),  
2239 Sunny Siu (MIT), Sanjay Sarma (MIT), and Dan Engels (MIT). In addition, we recognize  
2240 the contribution from Steve Rehling (P&G) on EPC to GTIN mapping.

2241 The following papers capture the contributions of these individuals:

- 2242 • Engels, D., Foley, J., Waldrop, J., Sarma, S. and Brock, D., "The Networked Physical  
2243 World: An Automated Identification Architecture"  
2244 2nd IEEE Workshop on Internet Applications (WIAPP '01),  
2245 (<http://csdl.computer.org/comp/proceedings/wiapp/2001/1137/00/11370076.pdf>)
- 2246 • Brock, David. "The Electronic Product Code (EPC), A Naming Scheme for Physical  
2247 Objects", 2001. (<http://www.autoidlabs.org/whitepapers/MIT-AUTOID-WH-002.pdf>)
- 2248 • Brock, David. "The Compact Electronic Product Code; A 64-bit Representation of the  
2249 Electronic Product Code", 2001. ([http://www.autoidlabs.com/whitepapers/MIT-](http://www.autoidlabs.com/whitepapers/MIT-AUTOID-WH-008.pdf)  
2250 [AUTOID-WH-008.pdf](http://www.autoidlabs.com/whitepapers/MIT-AUTOID-WH-008.pdf))

## 2251 **8 References**

- 2252 [EANUCCGS] "General EAN.UCC Specifications." Version 5.0, EAN International and  
2253 the Uniform Code Council, Inc<sup>TM</sup>, January 2004.
- 2254 [MIT-TR009] D. Engels, "The Use of the Electronic Product Code<sup>TM</sup>," MIT Auto-ID  
2255 Center Technical Report MIT-TR007, February 2003,  
2256 (<http://www.autoidlabs.com/whitepapers/mit-autoid-tr009.pdf>)
- 2257 [RFC2141] R. Moats, "URN Syntax," Internet Engineering Task Force Request for  
2258 Comments RFC-2141, May 1997, <http://www.ietf.org/rfc/rfc2141.txt>.
- 2259 [DOD Constructs] "United States Department of Defense Suppliers' Passive RFID  
2260 Information Guide," <http://www.dodrfid.org/supplierguide.htm>

2261

2262 **9 Appendix A: Encoding Scheme Summary Tables**

2263

<b>SGTIN Summary</b>						
<b>SGTIN-64</b>	<b>Header</b>	<b>Filter Value</b>	<b>Company Prefix Index</b>		<b>Item Reference</b>	<b>Serial Number</b>
	2 bits	3 bits	14 bits		20 bits	25 bits
	10 (Binary value)	(Refer to Table below for values)	16,383 (Max. decimal value)		9 - 1,048,575 (Max. decimal range*)	33,554,431 (Max. decimal value)
<b>SGTIN-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Item Reference</b>	<b>Serial Number</b>
	8	3	3	20-40	24 - 4	38
	0011 0000 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values))	999,999 – 999,999,999,999 (Max. decimal range**)	9,999,999 – 9 (Max .decimal range**)	274,877,906,943 (Max .decimal value)
<b>Filter Values (Non-normative)</b>		<b>SGTIN Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Item Reference and Indicator Digit</b>	
			<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digit</b>
All Others	000					
Retail Consumer Trade Item	001	0	40	12	4	1
Standard Trade Item Grouping	010	1	37	11	7	2
Single Shipping / Consumer Trade Item	011	2	34	10	10	3
Reserved	100	3	30	9	14	4
Reserved	101	4	27	8	17	5
Reserved	110	5	24	7	20	6
Reserved	111	6	20	6	24	7

2264

\*Range of Item Reference field varies with the length of the Company Prefix

2265

\*\*Range of Company Prefix and Item Reference fields vary according to the contents of the Partition field.

SSCC Summary						
SSCC-64	Header	Filter Value	Company Prefix Index		Serial Reference	
	8	3	14		39	
	0000 1000 (Binary value)	(Refer to Table below for values)	16,383 (Max. decimal value)		99,999 - 99,999,999,999 (Max. decimal range*)	
SSCC-96	Header	Filter Value	Partition	Company Prefix	Serial Reference	Unallocated
	8	3	3	20-40	38-18	24
	0011 0001 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	99,999,999,999 – 99,999 (Max. decimal range**)	[Not Used]
Filter Values (Non-normative)		SSCC Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Serial Reference and extension digit	
All Others	000		Bits	Digits	Bits	Digits
Undefined	001	0	40	12	18	5
Logistical / Shipping Unit	010	1	37	11	21	6
Reserved	011	2	34	10	24	7
Reserved	100	3	30	9	28	8
Reserved	101	4	27	8	31	9
Reserved	110	5	24	7	34	10
Reserved	111	6	20	6	38	11

2267 \*Range of Serial Reference field varies with the length of the Company Prefix

2268 \*\*Range of Company Prefix and Serial Reference fields vary according to the contents of the Partition field.

<b>SGLN Summary</b>						
<b>SGLN-64</b>	<b>Header</b>	<b>Filter Value</b>	<b>Company Prefix Index</b>		<b>Location Reference</b>	<b>Serial Number</b>
	8	3	14		20	19
	0000 1001 (Binary value)	(Refer to Table below for values)	16,383 (Max. decimal value)		999,999 - 0 (Max. decimal range*)	524,287 (Max. decimal value) [Not Used]
<b>SGLN-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Location Reference</b>	<b>Serial Number</b>
	8	3	3	20-40	21-1	41
	0011 0010 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	999,999 – 0 (Max. decimal range**)	2,199,023,255,551 (Max. decimal value) [Not Used]
<b>Filter Values (Non-normative)</b>		<b>SGLN Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Location Reference</b>	
			<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digit</b>
All Others	000					
Reserved	001	0	40	12	1	0
Reserved	010	1	37	11	4	1
Reserved	011	2	34	10	7	2
Reserved	100	3	30	9	11	3
Reserved	101	4	27	8	14	4
Reserved	110	5	24	7	17	5
Reserved	111	6	20	6	21	6

2270

\*Range of Location Reference field varies with the length of the Company Prefix

2271

\*\*Range of Company Prefix and Location Reference fields vary according to contents of the Partition field.

<b>GRAI Summary</b>						
<b>GRAI-64</b>	<b>Header</b>	<b>Filter Value</b>	<b>Company Prefix Index</b>		<b>Asset Type</b>	<b>Serial Number</b>
	8	3	14		20	19
	0000 1010 (Binary value)	(Refer to Table below for values)	16,383 (Max. decimal value)		999,999 - 0 (Max. decimal range*)	524,287 (Max. decimal capacity)
<b>GRAI-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Asset Type</b>	<b>Serial Number</b>
	8	3	3	20-40	24 - 4	38
	0011 0011 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 - 999,999,999,999 (Max. decimal range**)	999,999 - 0 (Max. decimal range**)	274,877,906,943 (Max. decimal value)
<b>Filter Values (Non-normative)</b>		<b>GRAI Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Asset Type</b>	
			<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digit</b>
All Others	000					
Reserved	001	0	40	12	4	0
Reserved	010	1	37	11	7	1
Reserved	011	2	34	10	10	2
Reserved	100	3	30	9	14	3
Reserved	101	4	27	8	17	4
Reserved	110	5	24	7	20	5
Reserved	111	6	20	6	24	6

2273 \*Range of Asset Type field varies with Company Prefix.

2274 \*\*Range of Company Prefix and Asset Type fields vary according to contents of the Partition field.

<b>GIAI Summary</b>						
<b>GIAI-64</b>	<b>Header</b>	<b>Filter Value</b>	<b>Company Prefix Index</b>		<b>Individual Asset Reference</b>	
	8	3	14		39	
	0000 1011 (Binary value)	(Refer to Table below for values)	16,383 (Max. decimal value)		549,755,813,887 (Max. decimal value)	
<b>GIAI-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Individual Asset Reference</b>	
	8	3	3	20-40	62-42	
	0011 0100 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range*)	4,611,686,018,427,387,903 – 4,398,046,511,103 (Max. decimal range*)	
<b>Filter Values (To be confirmed)</b>		<b>GIAI Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Individual Asset Reference</b>	
All Others	000		<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digits</b>
Reserved	001	0	40	12	42	12
Reserved	010	1	37	11	45	13
Reserved	011	2	34	10	48	14
Reserved	100	3	30	9	52	15
Reserved	101	4	27	8	55	16
Reserved	110	5	24	7	58	17
Reserved	111	6	20	6	62	18

2276

2277 \*Range of Company Prefix and Individual Asset Reference fields vary according to contents of the Partition  
 2278 field.

2279

2280 **10 Appendix B: EPC Header Values and Tag Identity**  
2281 **Lengths**

2282 With regards to tag identity lengths and EPC Header values: In the decoding process of a  
2283 single tag: Having knowledge of the identifier length during the signal decoding process  
2284 of the reader enables the reader to know when to stop trying to decode bit values.

2285 Knowing when to stop enables the readers to be more efficient in reading speed. For  
2286 example, if the same Header value is used at 64 and 96 bits, the reader, upon finding that  
2287 header value, must try to decode 96 bits. After decoding 96 bits, the reader must check  
2288 the CRC (Cyclic Redundancy Check error check code) against both the 64-bit and 96-bit  
2289 numbers it has decoded. If both error checks fail, the numbers are thrown away and the  
2290 tag reread. If one of the numbers passes the error check, then that is reported as the valid  
2291 number. Note that there is a non-zero, i.e., greater than zero but very small, probability  
2292 that an erroneous number can be reported in this process. If both numbers pass the error  
2293 check, then there is a problem. Note that there is a small probability that both a 64 bit

2294 EPC and 96-bit EPC whose first 64 bits are the same as the 64-bit EPC will have the  
2295 same CRC. Other measures would have to be taken to determine which of the two  
2296 numbers is valid (and perhaps both are). All of this slows down the reading process and  
2297 introduces potential errors in identified numbers (erroneous numbers may be reported)  
2298 and non-identified numbers (tags may be unread due to some of the above). These  
2299 problems are primarily evident while reading weakly replying tags, which are often the  
2300 tags furthest from the reader antenna and in noisy environments. Encoding the length  
2301 within the Header eliminates virtually all of the error probabilities above and those that  
2302 remain are reduced significantly in probability.

2303 In the decoding process of multiple tags responding: When multiple tags respond at the  
2304 same time their communications will overlap in time. Tags of the same length overlap  
2305 almost completely bit for bit when the same reader controls them. Tags of different  
2306 lengths will overlap almost completely over the first bits, but the longer tag will continue  
2307 communicating after the shorter tag has stopped. Tags of very strong communication  
2308 strength will mask tags responding with much weaker strength. The reader can use  
2309 communication signal strength as a determiner of when to stop looking to decode bits.  
2310 Tags of almost equal communication strength will tend to interfere almost completely  
2311 with one another over the first bits before the shorter tag stops. The reader can usually  
2312 detect these collisions, but not always when weak signals are trying to be pulled out of  
2313 noise, as is the case for the distant tags. When the tags reply with close, but not equal  
2314 strength, it may be possible to decode the stronger signal. When the short tag has the  
2315 stronger signal, it may be possible to decode the weaker longer tag signal without being  
2316 able to definitively say that a second tag is responding due to changes in signal strength.  
2317 These problems are primarily evident in weakly replying tags. Encoding the length in the  
2318 Header enables the reader to know when to stop pulling out the numbers, which enables it  
2319 to more efficiently determine the validity of the numbers.

2320 In the identification process: The reader can "select" what length tags it wishes to  
2321 communicate with. This eliminates the decoding problems encountered above, since all

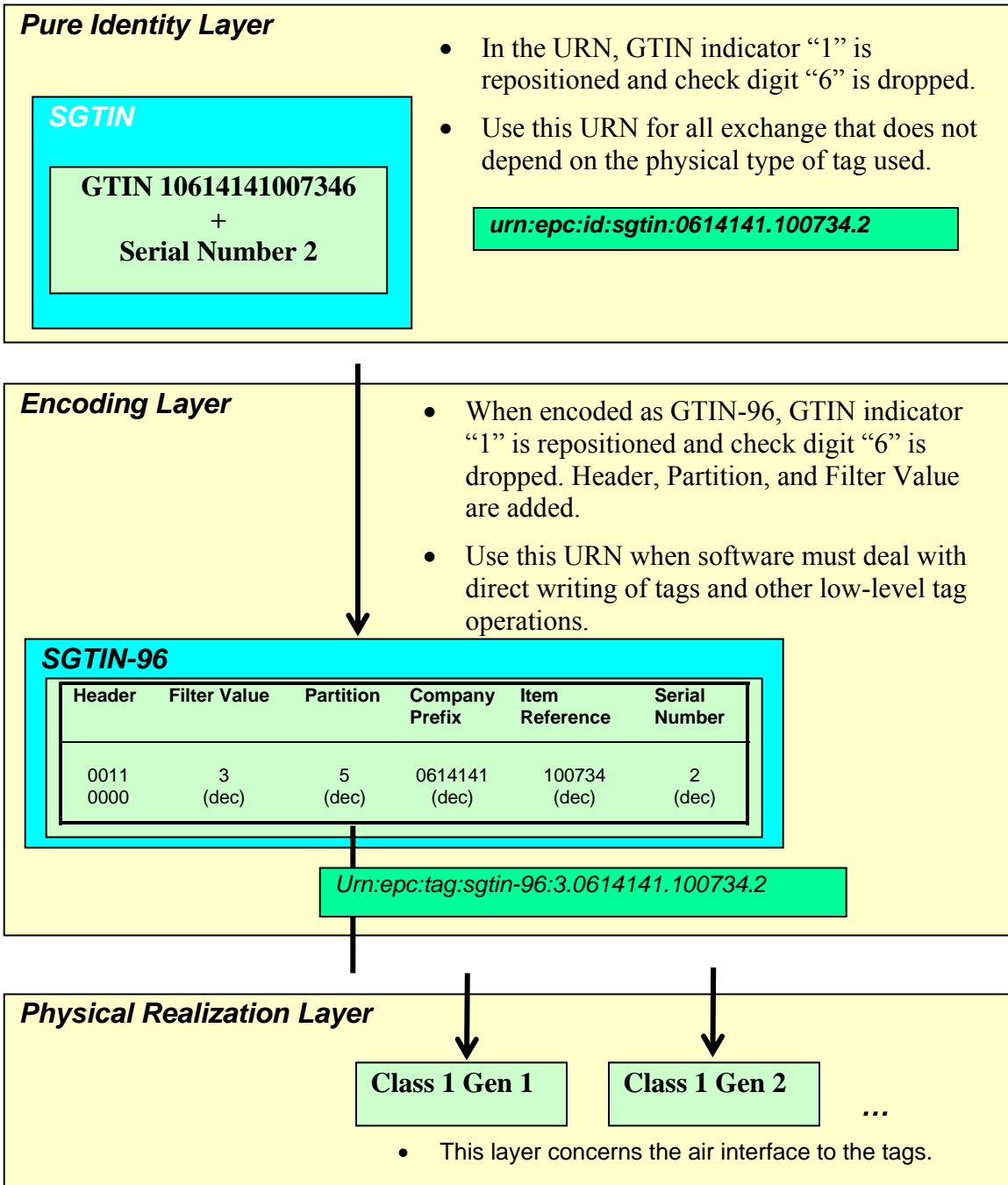


2322 communicating tags are of the same length and the reader knows what that length is a  
2323 priori. For efficiency reasons, a single selection for a length is preferred, but two can be  
2324 workable. More than two becomes very inefficient.

2325 The net effect of encoding the length within the Header is to reduce the probabilities of  
2326 error in the decoding process and to increase the efficiency of the identification process.

2327 **11 Appendix C: Example of a Specific Trade Item**  
 2328 **(SGTIN)**

2329 This section presents an example of a specific trade item using SGTIN (Serialized GTIN).  
 2330 Each representation serves a distinct purpose in the software stack. Generally,, the  
 2331 highest applicable level should be used. The GTIN used in the example is  
 2332 10614141007346.



2333



2334

2335

2336

2337

	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
SGTIN-96	8 bits	3 bits	3 bits	24 bits	20 bits	38 bits
	0011 0000 (Binary value)	3 (Decimal value)	5 (Decimal value)	0614141 (Decimal value)	100734 (Decimal value)	2 (Decimal value)

2338

2339

2340

2341

2342

2343

2344

2345

2346

2347

2348

2349

2350

Explanation of SGTIN Filter Values (non-normative).

2351

2352

2353

2354

2355

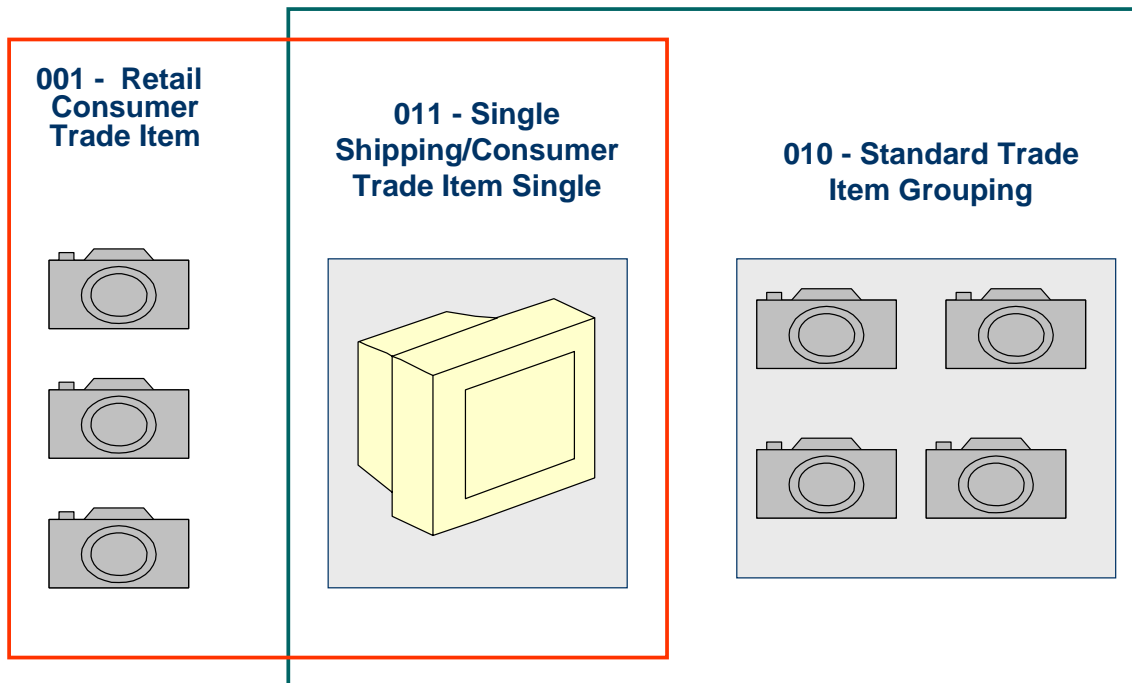
SGTINs can be assigned at several levels, including: item, inner pack, case, and pallet. RFID can read through cardboard, and reading un-needed tags can slow us down, so Filter Values are used to “filter in” desired tags, or “filter out” unwanted tags. Filter values are used within the key type (i.e. SGTIN). While it is possible that filter values for several levels of packaging may be defined in the future, it was decided to use a

2356 minimum of values for now until the community gains more practical experience in their  
2357 use. Therefore the three major categories of SGTIN filter values can be thought of in the  
2358 following high level terms:

- 2359 • Single Unit: A Retail Consumer Trade Item
- 2360 • Not-a-single unit: A Standard Trade Item Grouping
- 2361 • Items that could be included in both categories: For example, a Single Shipping  
2362 container that contains a Single Consumer Trade Item

2363

## Three Filter Values



2364

2365

2366

### 12 Appendix D: Decimal values of powers of 2 Table

2367

n	(2^n) <sub>10</sub>	n	(2^n) <sub>10</sub>
0	1	33	8,589,934,592
1	2	34	17,179,869,184
2	4	35	34,359,738,368
3	8	36	68,719,476,736
4	16	37	137,438,953,472
5	32	38	274,877,906,944
6	64	39	549,755,813,888
7	128	40	1,099,511,627,776
8	256	41	2,199,023,255,552
9	512	42	4,398,046,511,104
10	1,024	43	8,796,093,022,208
11	2,048	44	17,592,186,044,416
12	4,096	45	35,184,372,088,832
13	8,192	46	70,368,744,177,664
14	16,384	47	140,737,488,355,328
15	32,768	48	281,474,976,710,656
16	65,536	49	562,949,953,421,312
17	131,072	50	1,125,899,906,842,624
18	262,144	51	2,251,799,813,685,248
19	524,288	52	4,503,599,627,370,496
20	1,048,576	53	9,007,199,254,740,992
21	2,097,152	54	18,014,398,509,481,984
22	4,194,304	55	36,028,797,018,963,968
23	8,388,608	56	72,057,594,037,927,936
24	16,777,216	57	144,115,188,075,855,872
25	33,554,432	58	288,230,376,151,711,744
26	67,108,864	59	576,460,752,303,423,488
27	143,217,728	60	1,152,921,504,606,846,976
28	268,435,456	61	2,305,843,009,213,693,952
29	536,870,912	62	4,611,686,018,427,387,904
30	1,073,741,824	63	9,223,372,036,854,775,808
31	2,147,483,648	64	18,446,744,073,709,551,616
32	4,294,967,296		

2368

2369 **13 Appendix E: List of Abbreviations**

2370

BAG	Business Action Group
EPC	Electronic Product Code
EPCIS	EPC Information Services
GIAI	Global Individual Asset Identifier
GID	General Identifier
GLN	Global Location Number
GRAI	Global Returnable Asset Identifier
GTIN	Global Trade Item Number
HAG	Hardware Action Group
ONS	Object Naming Service
RFID	Radio Frequency Identification
SAG	Software Action Group
SGLN	Serialized Global Location Number
SSCC	Serial Shipping Container Code
URI	Uniform Resource Identifier
URN	Uniform Resource Name

2371

2372

2373 **14 Appendix F: General EAN.UCC Specifications**

2374 (Section 3.0 Definition of Element Strings and Section 3.7 EPCglobal Tag Data  
2375 Standard.)

2376 This section provides EAN.UCC approval of this version of the EPCglobal® Tag Data  
2377 Standard with the following EAN.UCC Application Identifier definition restrictions:

2378 Companies should use the EAN.UCC specifications to define the applicable fields in  
2379 databases and other ICT-systems.

2380 For EAN.UCC use of EPC 64-bit tags, the following applies:

- 64-bit tag application is limited to 16,383 EAN.UCC Company Prefixes and therefore EAN.UCC EPCglobal implementation strategies will focus on tag capacity that can accommodate all EAN.UCC member companies. The 64-bit tag will be approved for use by EAN.UCC member companies with the restrictions that follow:

2381 • **AI (00) SSCC** (no restrictions)

2382 • **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is  
2383 restricted to permit assignment of 33,554,431 numeric-only serial numbers.

2384 • **AI (41n) GLN + AI (21) Serial Number:** The Tag Data Standard V1.1 R1.23 is approved  
2385 with a complete restriction on GLN serialization because this question has not been  
2386 resolved by GSMP at this time.

2387 • **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier  
2388 definition is restricted to permit assignment of 524,288 numeric-only serial numbers and  
2389 the serial number element is mandatory.

2390 • **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier  
2391 definition is restricted to permit assignment of 549,755,813,888 numeric-only serial  
2392 numbers.

2393 For EAN.UCC use of EPC96-bit tags, the following applies:

2394 • **AI (00) SSCC** (no restrictions)

2395 • **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is  
2396 restricted to permit assignment of 274,877,906,943 numeric-only serial numbers)

2397 • **AI (41n) GLN + AI (21) Serial Number:** The Tag Data Standard V1.1 R1.23 is approved  
2398 with a complete restriction on GLN serialization because this question has not been  
2399 resolved by GSMP at this time.

2400 • **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier  
2401 definition is restricted to permit assignment of 274,877,906,943 numeric-only serial  
2402 numbers and the serial number element is mandatory.

2403 • **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier  
2404 definition is restricted to permit assignment of 4,611,686,018,427,387,904 numeric-only  
2405 serial numbers.

2406