



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

EPC™ Tag Data Standards Version 1.1 Rev.1.26

Standard Specification
19 November 2004

25 **DOCUMENT HISTORY**

26

Document Number:	1.1
Document Version:	1.26
Document Date :	2004-11-19

27
28
29
30 **Document Summary**

31

Document Title:	Tag Data Specification
Owner:	Tag Data Standard Work Group Chairperson: Yuichiro Hanawa, Mitsui & Co.
Status:	(<i>check one box</i>) <input type="checkbox"/> DRAFT <input checked="" type="checkbox"/> Approved

32
33 **Document Change History**

34

Date of Change	Version	Reason for Change	Summary of Change
03-31-2004	1.24	Update errata	Comments and errata identified during public review
08-12-2004	1.25	Update errata	Further errata identified after release of v1.24 – especially inconsistencies regarding SSCC-96 and GRAI-96 partition tables
09-16-2004	1.25	Update errata	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
09-30-2004	1.26		Added clarification to restrictions to serial numbers in SGTIN, GRAI, GIAI

36 **Abstract**

37 This document defines the EPC Tag Data Standards. These standards define completely
38 that portion of EPC tag data that is standardized, including how that data is encoded on
39 the EPC tag itself (i.e. the EPC Tag Encodings), as well as how it is encoded for use in
40 the information systems layers of the EPC Systems Network (i.e. the EPC URI or
41 Uniform Resource Identifier Encodings).

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

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

- 50 1. URIs for pure identities, sometimes called “canonical forms.” These contain only
51 the unique information that identifies a specific physical object, and are
52 independent of tag encodings.
- 53 2. URIs that represent specific tag encodings. These are used in software
54 applications where the encoding scheme is relevant, as when commanding
55 software to write a tag.
- 56 3. URIs that represent patterns, or sets of EPCs. These are used when instructing
57 software how to filter tag data.
- 58 4. URIs that represent raw tag information, generally used only for error reporting
59 purposes.

60

61 **Status of this document**

62 This section describes the status of this document at the time of its publication. Other
63 documents may supersede this document. The latest status of this document series is
64 maintained at EPCglobal. This document is the ratified specification named Tag Data
65 Standards Version 1.1 Rev.1.26.

66 Comments on this document should be sent to EPCglobal at epcinfo@epcglobalinc.org.

67 **Changes from Previous Versions**

68 Version 1.1, as the first formally specified version, serves as the basis for assignment and
69 use of EPC numbers in standard, open systems applications. Previous versions, consisting
70 of technical reports and working drafts, recommended certain headers, tag lengths, and
71 EPC data structures. Many of these constructs have been modified in the development of

72 Version 1.1, and are generally not preserved for standard usage. Specifically, Version 1.1
73 supersedes all previous definitions of EPC Tag Data Standards.

74

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

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

93

95 **Table of Contents**

96	1	Introduction	8
97	2	Identity Concepts.....	9
98	2.1	Pure Identities	10
99	2.1.1	General Types	10
100	2.1.2	EAN.UCC System Identity Types	11
101	2.1.2.1	Serialized Global Trade Item Number (SGTIN).....	12
102	2.1.2.2	Serial Shipping Container Code (SSCC)	13
103	2.1.2.3	Serialized Global Location Number (SGLN).....	14
104	2.1.2.4	Global Returnable Asset Identifier (GRAI)	15
105	2.1.2.5	Global Individual Asset Identifier (GIAI).....	15
106	3	EPC Tag Bit-level Encodings	16
107	3.1	Headers	16
108	3.2	Notational Conventions	18
109	3.3	General Identifier (GID-96).....	19
110	3.3.1.1	GID-96 Encoding Procedure	20
111	3.3.1.2	GID-96 Decoding Procedure.....	20
112	3.4	Serialized Global Trade Item Number (SGTIN)	21
113	3.4.1	SGTIN-64	21
114	3.4.1.1	SGTIN-64 Encoding Procedure	23
115	3.4.1.2	SGTIN-64 Decoding Procedure	23
116	3.4.2	SGTIN-96	24
117	3.4.2.1	SGTIN-96 Encoding Procedure	26
118	3.4.2.2	SGTIN-96 Decoding Procedure	27
119	3.5	Serial Shipping Container Code (SSCC).....	28
120	3.5.1	SSCC-64	28
121	3.5.1.1	SSCC-64 Encoding Procedure	29
122	3.5.1.2	SSCC-64 Decoding Procedure	30
123	3.5.2	SSCC-96	31

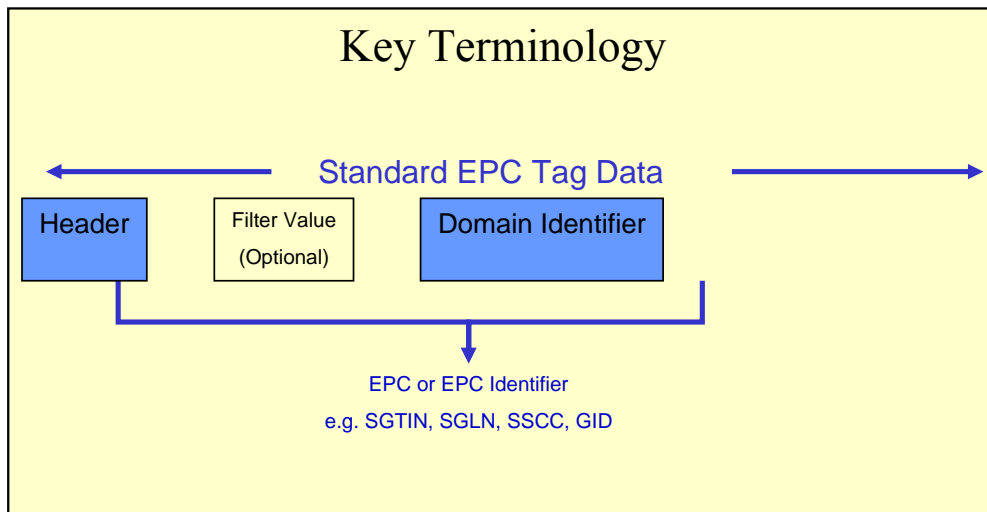
124	3.5.2.1	SSCC-96 Encoding Procedure	32	
125	3.5.2.2	SSCC-96 Decoding Procedure	33	
126	3.6	Serialized Global Location Number (SGLN).....	33	
127	3.6.1	SGLN-64.....	34	
128	3.6.1.1	SGLN-64 Encoding Procedure.....	35	
129	3.6.1.2	SGLN-64 Decoding Procedure	35	Deleted: 36
130	3.6.2	SGLN-96.....	36	
131	3.6.2.1	SGLN-96 Encoding Procedure.....	38	
132	3.6.2.2	SGLN-96 Decoding Procedure	38	
133	3.7	Global Returnable Asset Identifier (GRAI).....	39	
134	3.7.1	GRAI-64	39	
135	3.7.1.1	GRAI-64 Encoding Procedure	41	
136	3.7.1.2	GRAI-64 Decoding Procedure	41	Deleted: 42
137	3.7.2	GRAI-96	42	
138	3.7.2.1	GRAI-96 Encoding Procedure	43	Deleted: 44
139	3.7.2.2	GRAI-96 Decoding Procedure.....	44	
140	3.8	Global Individual Asset Identifier (GIAI).....	45	
141	3.8.1	GIAI-64.....	45	Deleted: 46
142	3.8.1.1	GIAI-64 Encoding Procedure.....	47	
143	3.8.1.2	GIAI-64 Decoding Procedure	47	Deleted: 48
144	3.8.2	GIAI-96.....	48	
145	3.8.2.1	GIAI-96 Encoding Procedure.....	49	
146	3.8.2.2	GIAI-96 Decoding Procedure	50	
147	4	URI Representation	50	Deleted: 51
148	4.1	URI Forms for Pure Identities	51	
149	4.2	URI Forms for Related Data Types.....	53	
150	4.2.1	URIs for EPC Tags	53	
151	4.2.2	URIs for Raw Bit Strings Arising From Invalid Tags.....	54	
152	4.2.3	URIs for EPC Patterns	55	
153	4.3	Syntax	55	
154	4.3.1	Common Grammar Elements	55	Deleted: 56
155	4.3.2	EPCGID-URI.....	56	

156	4.3.3	SGTIN-URI.....	56	Deleted: 57
157	4.3.4	SSCC-URI.....	57	
158	4.3.5	SGLN-URI.....	57	
159	4.3.6	GRAI-URI.....	57	
160	4.3.7	GIAI-URI.....	57	Deleted: 58
161	4.3.8	EPC Tag URI.....	58	
162	4.3.9	Raw Tag URI.....	58	
163	4.3.10	EPC Pattern URI.....	58	Deleted: 59
164	4.3.11	Summary (non-normative).....	59	
165	5	Translation between EPC-URI and Other EPC Representations.....	61	
166	6	Semantics of EPC Pattern URIs.....	67	
167	7	Background Information.....	68	
168	8	References.....	69	Deleted: 70
169	9	Appendix A: Encoding Scheme Summary Tables.....	71	
170	10	Appendix B: EPC Header Values and Tag Identity Lengths.....	76	
171	11	Appendix C: Example of a Specific Trade Item (SGTIN).....	78	
172	12	Appendix D: Binary Digit Capacity Tables.....	81	Deleted: 80
173	13	Appendix E: List of Abbreviations.....	82	Deleted: 81
174	14	Appendix F: General EAN.UCC Specifications.....	83	Deleted: 82
175				

176 **1 Introduction**

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

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



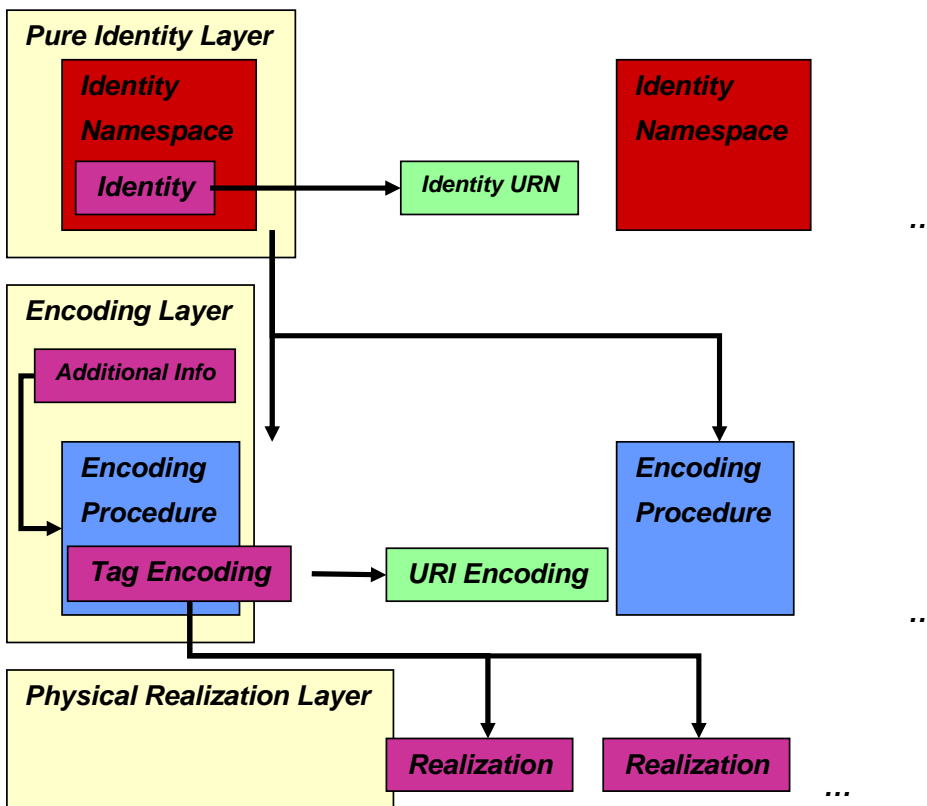
194
195 **Figure A.** EPC Terminology.

196
197 In this version of the EPC – EPC Version 1.1 – the specific coding schemes include a
198 General Identifier (GID), a serialized version of the EAN.UCC Global Trade Item
199 Number (GTIN®), the EAN.UCC Serial Shipping Container Code (SSCC®), the

200 EAN.UCC Global Location Number (GLN®), the EAN.UCC Global Returnable Asset
 201 Identifier (GRAI®), and the EAN.UCC Global Individual Asset Identifier (GIAI®).
 202 In the following sections, we will describe the structure and organization of the EPC and
 203 provide illustrations to show its recommended use.
 204 The EPCglobal Tag Data Standard V1.1 R1.26 has been approved by EAN.UCC with the
 205 restrictions outlined in the General EAN.UCC Specifications Section 3.7, which is
 206 excerpted into Tag Data Standard Appendix F.
 207 The latest version of this specification can be found online at www.epcglobalinc.org.

208 **2 Identity Concepts**

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



213

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

215 Pure identity -- the identity associated with a specific physical or logical entity,
216 independent of any particular encoding vehicle such as an RF tag, bar code or database
217 field. As such, a pure identity is an abstract name or number used to identify an entity. A
218 pure identity consists of the information required to uniquely identify a specific entity,
219 and no more. Identity URI – a representation of a pure identity as a Uniform Resource
220 Identifier (URI). A URI is a character string representation that is commonly used to
221 exchange identity data between software components of a larger system.

222 Encoding -- a pure identity, together with additional information such as filter value,
223 rendered into a specific syntax (typically consisting of value fields of specific sizes). A
224 given pure identity may have a number of possible encodings, such as a Barcode
225 Encoding, various Tag Encodings, and various URI Encodings. Encodings may also
226 incorporate additional data besides the identity (such as the Filter Value used in some
227 encodings), in which case the encoding scheme specifies what additional data it can hold.

228 Physical Realization of an Encoding -- an encoding rendered in a concrete
229 implementation suitable for a particular machine-readable form, such as a specific kind of
230 RF tag or specific database field. A given encoding may have a number of possible
231 physical realizations.

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

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

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

247 **2.1 Pure Identities**

248 This section defines the pure identity types for which this document specifies encoding
249 schemes.

250 **2.1.1 General Types**

251 This version of the EPC Tag Data Standards defines one general identity type. The
252 *General Identifier (GID-96)* is independent of any known, existing specifications or
253 identity schemes. The General Identifier is composed of three fields - the *General*
254 *Manager Number*, *Object Class* and *Serial Number*. Encodings of the GID include a
255 fourth field, the header, to guarantee uniqueness in the EPC namespace.

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

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

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

268 **2.1.2 EAN.UCC System Identity Types**

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

271 EAN.UCC System codes have a common structure, consisting of a fixed number of
272 decimal digits that encode the identity, plus one additional “check digit” which is
273 computed algorithmically from the other digits. Within the non-check digits, there is an
274 implicit division into two fields: a Company Prefix assigned by EAN or UCC to a
275 managing entity, and the remaining digits, which are assigned by the managing entity.
276 (The digits apart from the Company Prefix are called by a different name by each of the
277 EAN.UCC System codes.) The number of decimal digits in the Company Prefix varies
278 from 6 to 12 depending on the particular Company Prefix assigned. The number of
279 remaining digits therefore varies inversely so that the total number of digits is fixed for a
280 particular EAN.UCC System code type.

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

292 EPC encodings carry an explicit division between the Company Prefix and the remaining
293 digits, with each individually encoded into binary. Hence, converting from the traditional
294 decimal representation of an EAN.UCC System code and an EPC encoding requires
295 independent knowledge of the length of the Company Prefix.

296 EPC encodings do not include the check digit. Hence, converting from an EPC encoding
297 to a traditional decimal representation of a code requires that the check digit be
298 recalculated from the other digits.

299 2.1.2.1 Serialized Global Trade Item Number (SGTIN)

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

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

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

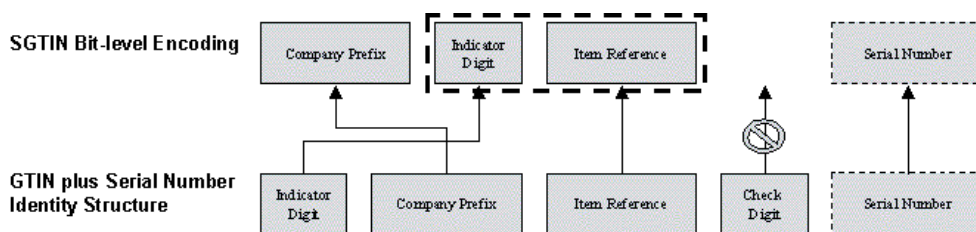
314 The SGTIN consists of the following information elements:

315 The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company
316 Prefix is the same as the Company Prefix digits within an EAN.UCC GTIN decimal code.

317 The *Item Reference*, assigned by the managing entity to a particular object class. The
318 Item Reference for the purposes of EPC encoding is derived from the GTIN by
319 concatenating the Indicator Digit of the GTIN and the Item Reference digits, and treating
320 the result as a single integer.

321 The *Serial Number*, assigned by the managing entity to an individual object. The serial
322 number is not part of the GTIN code, but is formally a part of the SGTIN.

323



324

325

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

328 The SGTIN is not explicitly defined in the EAN.UCC General Specifications. However,
329 it may be considered equivalent to a UCC/EAN-128 bar code that contains both a GTIN

330 (Application Identifier 01) and a serial number (Application Identifier 21). Serial
331 numbers in AI 21 consist of one to twenty characters, where each character can be a digit,
332 uppercase or lowercase letter, or one of a number of allowed punctuation characters. The
333 complete AI 21 syntax is supported by the pure identity URI syntax specified in
334 Section 4.3.3.

335 When representing serial numbers in 64- and 96-bit tags, however, only a subset of the
336 serial number allowed in the General EAN.UCC Specifications for Application Identifier
337 21 are permitted. Specifically, the permitted serial numbers are those consisting of one or
338 more digits characters, with no leading zeros, and whose value when considered as an
339 integer fits within the range restrictions of the 64- and 96-bit tag encodings.

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

344 *Explanation (non-normative): The restrictions are necessary for 64- and 96-bit tags in*
345 *order for serial numbers to fit within the small number of bits we have available. So we*
346 *restrict the range, and also disallow alphabetic characters. The reason we also forbid*
347 *leading zeros is that on these tags we're encoding the serial number value by considering*
348 *it to be a decimal integer then encoding the integer value in binary. By considering it to*
349 *be a decimal integer, we can't distinguish between "00034", "034", or "34" (for example)*
350 *-- they all have the same value when considered as an integer rather than a character*
351 *string. In order to insure that every encoded value can be decoded uniquely, we*
352 *arbitrarily say that serial numbers can't have leading zeros. Then, when we see the bits*
353 *0000000000000000000010010 on the tag, we decode the serial number as "34" (not*
354 *"034" or "00034").*

355 **2.1.2.2 Serial Shipping Container Code (SSCC)**

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

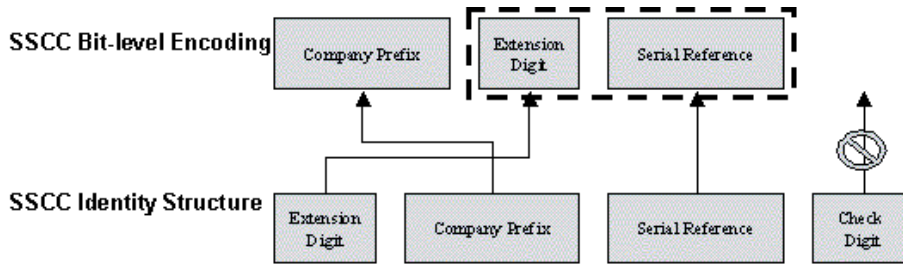
360 *Note that many applications of SSCC have historically included the Application Identifier*
361 *(00) in the SSCC identifier field when stored in a database. This is not a standard*
362 *requirement, but a widespread practice. The Application Identifier is a sort of header*
363 *used in bar code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

366 The SSCC consists of the following information elements:

367 The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company
368 Prefix is the same as the Company Prefix digits within an EAN.UCC SSCC decimal code.

369 The *Serial Reference*, assigned uniquely by the managing entity to a specific shipping
370 unit. The Serial Reference for the purposes of EPC encoding is derived from the SSCC

371 by concatenating the Extension Digit of the SSCC and the Serial Reference digits, and
 372 treating the result as a single integer.
 373



374
 375 **Figure D.** How the parts of the decimal SSCC are extracted and rearranged for
 376 encoding.

377 **2.1.2.3 Serialized Global Location Number (SGLN)**

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

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

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

387 The SGLN consists of the following information elements:

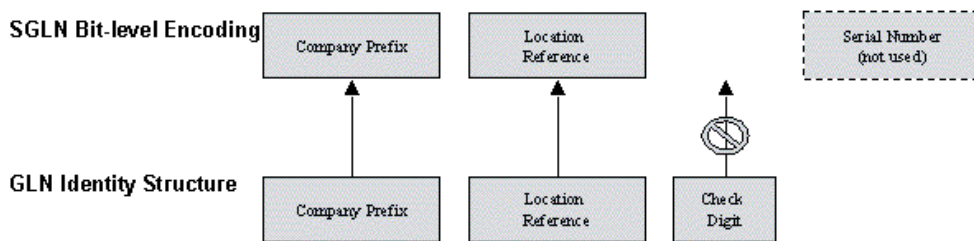
388 The *Company Prefix*, assigned by EAN or UCC to a managing entity. The Company
 389 Prefix is the same as the Company Prefix digits within an EAN.UCC GLN decimal code.

390 The *Location Reference*, assigned uniquely by the managing entity to an aggregate or
 391 specific physical location.

392 The *Serial Number*, assigned by the managing entity to an individual unique location.

- 393 ➤ The serial number should not be used until specified by the EAN.UCC General
 394 Specifications .

395



396

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

398 **2.1.2.4 Global Returnable Asset Identifier (GRAI)**

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

403

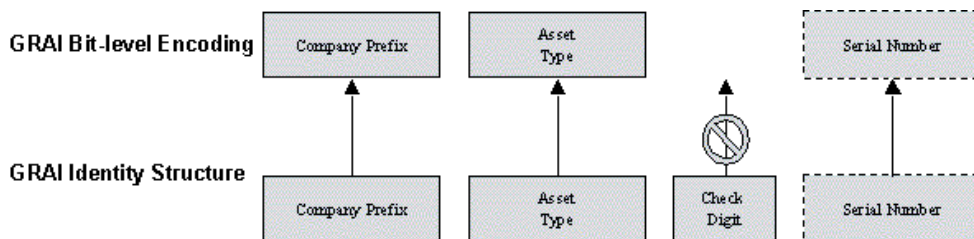
404 The GRAI consists of the following information elements:

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 GRAI decimal code.

407 The *Asset Type*, assigned by the managing entity to a particular class of asset.

408 The *Serial Number*, assigned by the managing entity to an individual object. The EPC
409 representation is only capable of representing a subset of Serial Numbers allowed in the
410 General EAN.UCC Specifications. Specifically, only those Serial Numbers consisting of
411 one or more digits, with no leading zeros, are permitted [see Appendix F for details].

412



413

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

415 **2.1.2.5 Global Individual Asset Identifier (GIAI)**

416 The Global Individual Asset Identifier (GIAI) is defined by the General EAN.UCC
417 Specifications. Unlike the GTIN, the GIAI 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.

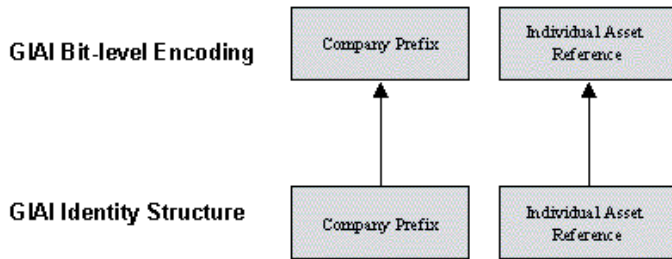
420

421 The GIAI 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 GIAI decimal code.

424 The *Individual Asset Reference*, assigned uniquely by the managing entity to a specific
425 asset. The EPC representation is only capable of representing a subset of Individual Asset
426 References allowed in the General EAN.UCC Specifications. Specifically, only those

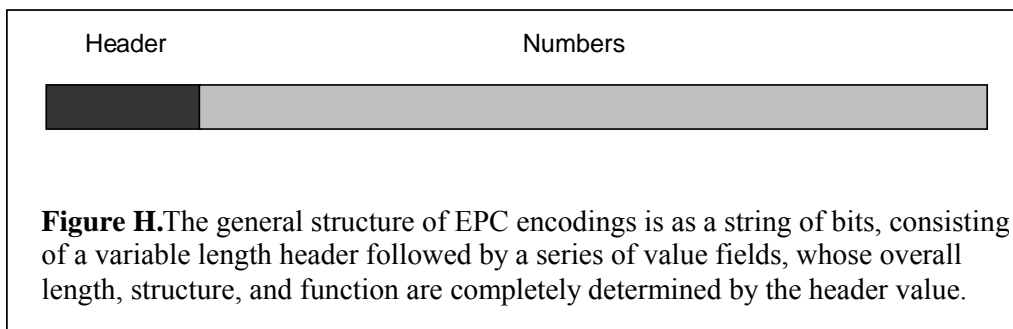
427 Individual Asset References consisting of one or more digits, with no leading zeros, are
 428 permitted.
 429



430
 431 **Figure G.** How the parts of the decimal GIAI are extracted and rearranged for encoding.

432 3 EPC Tag Bit-level Encodings

433 The general structure of EPC encodings on a tag is as a string of bits (i.e., a binary
 434 representation), consisting of a tiered, variable length header followed by a series of
 435 numeric fields (Figure H) whose overall length, structure, and function are completely
 436 determined by the header value.
 437



438 3.1 Headers

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

447 *Explanation (non-normative): The tiered scheme is designed to simplify the Header*
 448 *processing required by the Reader in order to determine the tag data format, particularly*

449 *the location of the Filter Value, while attempting to conserve bits for data values in the*
 450 *64-bit tag. In the not-too-distant future, we expect to be able to “reclaim” the 2-bit tier*
 451 *when 64-bit tags are no longer needed, thereby expanding the 8-bit Header from 63*
 452 *possible values to 255.*

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

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

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

469 Eleven encoding schemes have been defined in this version of the EPC Tag Data
 470 Standard, as shown in Table 1 below.

Header Value (binary)	Tag Length (bits)	EPC Encoding Scheme
01	64	[Reserved 64-bit scheme]
10	64	SGTIN-64
11	64	[Reserved 64-bit scheme]
<i>0000 0001</i>	<i>na</i>	<i>[1 reserved scheme]</i>
<i>0000 001x</i>	<i>na</i>	<i>[2 reserved schemes]</i>
<i>0000 01xx</i>	<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	64	[4 reserved 64-bit schemes]
...		
0000 1111		

Header Value (binary)	Tag Length (bits)	EPC Encoding Scheme
0001 0000 ... 0010 1111	Na	[32 reserved schemes]
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
0011 0101	96	GID-96
0011 0110 ... 0011 1111	96	[10 reserved 96-bit schemes]
0000 0000 ...		[reserved for future headers longer than 8 bits]

Table 1. Electronic Product Code Headers

471

472

3.2 Notational Conventions

473

474 In the remainder of this section, tag-encoding schemes are depicted using the following
475 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)	8 (Decimal capacity)	16,383 (Decimal capacity)	9 -1,048,575 (Decimal capacity*)	33,554,431 (Decimal capacity)

*Capacity of Item Reference field varies with the length of the Company Prefix

476

Table 2. Example of Notation Conventions.

477

478

479 The first column of the table gives the formal name for the encoding. The remaining
480 columns specify the layout of each field within the encoding. The field in the leftmost

481 column occupies the most significant bits of the encoding (this is always the header field),
 482 and the field in the rightmost column occupies the least significant bits. Each field is a
 483 non-negative integer, encoded into binary using a specified number of bits. Any unused
 484 bits (i.e., bits not required by a defined field) are explicitly indicated in the table, so that
 485 the columns in the table are concatenated with no gaps to form the complete binary
 486 encoding.

487 Reading down each column, the table gives the formal name of the field, the number of
 488 bits used to encode the field's value, and the number of possible values that are permitted
 489 within that field. The number of possible values in the field can be either a specified limit,
 490 or simply two to the power of the number of bits in the field.

491 In some cases, the number of possible values in one field depends on the specific value
 492 assigned to another field. In such cases, a range of decimal capacity is shown. In the
 493 example above, the decimal capacity for the Item Reference field depends on the length
 494 of the Company Prefix field; hence the decimal capacity is shown as a range. Where a
 495 field must contain a specific value (as in the Header field), the last row of the table
 496 specifies the specific value rather than the number of possible values.

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

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

502 3.3 General Identifier (GID-96)

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

508

	Header	General Manager Number	Object Class	Serial Number
GID-96	8	28	24	36
	0011 0101 (Binary value)	268,435,456 (Decimal capacity)	16,777,216 (Decimal capacity)	68,719,476,736 (Decimal capacity)

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

511

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

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

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

523 **3.3.1.1 GID-96 Encoding Procedure**

524 The following procedure creates a GID-96 encoding.

525 Given:

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

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

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

529 Procedure:

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

533 **3.3.1.2 GID-96 Decoding Procedure**

534 Given:

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

537 Yields:

538 An General Manager Number

539 An Object Class

540 A Serial Number

541 Procedure:

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

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

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

545 **3.4 Serialized Global Trade Item Number (SGTIN)**

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

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

559
 560

561 **3.4.1 SGTIN-64**

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

564
 565
 566

	Header	Filter Value	Company Prefix Index	Item Reference	Serial Number
SGTIN-64	2	3	14	20	25
	10 (Binary value)	8 (Decimal capacity)	16,383 (Decimal capacity)	9 -1,048,575 (Decimal capacity*)	33,554,431 (Decimal capacity)

567 *Capacity of Item Reference field varies with the length of the Company Prefix

568 **Table 4.** The EPC SGTIN-64 bit allocation, header, and decimal capacity.

569

570 *Header* is 2 bits, with a binary value of 10.

571 *Filter Value* is not part of the SGTIN pure identity, but is additional data that is used for
 572 fast filtering and pre-selection of basic logistics types, such as items, inner packs, cases
 573 and pallets. The Filter Values for 64-bit and 96-bit SGTIN are the same. The normative
 574 specifications for Filter Values are specified in Table 5. The value of 000 means
 575 “unspecified”. That is, a filter value of 000 means that the tag provides no information as

576 to the logistics type of the object to which the tag is affixed. In particular, tags
 577 conforming to earlier versions of this specification, in which 000 was the only value
 578 approved for use, will have filter equal to 000. In general, filter 000 should not be used
 579 for SGTIN EPC tags intended for use in the supply chain following ratification of this
 580 standard. A Standard Trade Item grouping represents all levels of packaging for
 581 logistical units. The Single Shipping / Consumer Trade item type should be used when
 582 the individual item is also the logistical unit (e.g. Large screen television, Bicycle).

583

Type	Binary Value
Unspecified	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

584

Table 5. SGTIN Filter Values .

585

586 *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this field is
 587 not the Company Prefix itself, but rather an index into a table that provides the Company
 588 Prefix as well as an indication of the Company Prefix's length. The means by which
 589 hardware or software may obtain the contents of the translation table is specified in
 590 [Translation of 64-bit Tag Encoding Company Prefix Indices Into EAN.UCC Company
 591 Prefixes].

592 *Item Reference* encodes the GTIN Item Reference number and Indicator Digit. The
 593 Indicator Digit is combined with the Item Reference field in the following manner:
 594 Leading zeros on the item reference are significant. Put the Indicator Digit in the leftmost
 595 position available within the field. *For instance, 00235 is different than 235. With the*
 596 *indicator digit of 1, the combination with 00235 is 100235.* The resulting combination is
 597 treated as a single integer, and encoded into binary to form the Item Reference field.

598 *Serial Number* contains a serial number. The SGTIN-64 encoding is only capable of
 599 representing integer-valued serial numbers with limited range. Other EAN.UCC
 600 specifications permit a broader range of serial numbers. In particular, the EAN-128
 601 barcode symbology provides for a 20-character alphanumeric serial number to be
 602 associated with a GTIN using Application Identifier (AI) 21 [EANUCCGS]. It is
 603 possible to convert between the serial numbers in the SGTIN-64 tag encoding and the
 604 serial numbers in AI 21 barcodes under certain conditions. Specifically, such

Formatted: Bulleted + Level: 1 +
 Aligned at: 18 pt + Tab after: 36 pt
 + Indent at: 36 pt, Tabs: 18 pt, Left

605 interconversion is possible when the alphanumeric serial number in AI 21 happens to
606 consist only of digit characters, with no leading zeros, and whose value when interpreted
607 as an integer falls within the range limitations of the SGTIN-64 tag encoding. These
608 considerations are reflected in the encoding and decoding procedures below.

609 **3.4.1.1 SGTIN-64 Encoding Procedure**

610 The following procedure creates an SGTIN-64 encoding.

611 Given:

612 An EAN.UCC GTIN-14 consisting of digits $d_1d_2\dots d_{14}$

613 The length L of the company prefix portion of the GTIN

614 A Serial Number S where $0 \leq S < 2^{25}$, or an UCC/EAN-128 Application Identifier 21
615 consisting of characters $s_1s_2\dots s_K$.

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

617 Procedure:

- 618 1. Extract the EAN.UCC Company Prefix $d_2d_3\dots d_{(L+1)}$
- 619 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table
620 to obtain the corresponding Company Prefix Index, C . If the Company Prefix was not
621 found in the Company Prefix Translation Table, stop: this GTIN cannot be encoded in the
622 SGTIN-64 encoding.
- 623 3. Construct the Item Reference by concatenating digits $d_1d_{(L+2)}d_{(L+3)}\dots d_{13}$ and
624 considering the result to be a decimal integer, I . If $I \geq 2^{20}$, stop: this GTIN cannot be
625 encoded in the SGTIN-64 encoding.
- 626 4. When the Serial Number is provided directly as an integer S where $0 \leq S < 2^{25}$,
627 proceed to Step 5. Otherwise, when the Serial Number is provided as an UCC/EAN-128
628 Application Identifier 21 consisting of characters $s_1s_2\dots s_K$, construct the Serial Number
629 by concatenating digits $s_1s_2\dots s_K$. If any of these characters is not a digit, stop: this Serial
630 Number cannot be encoded in the SGTIN-64 encoding. Also, if $K > 1$ and $s_1 = 0$, stop:
631 this Serial Number cannot be encoded in the SGTIN-64 encoding (because leading zeros
632 are not permitted except in the case where the Serial Number consists of a single zero
633 digit). Otherwise, consider the result to be a decimal integer, S . If $S \geq 2^{25}$, stop: this
634 Serial Number cannot be encoded in the SGTIN-64 encoding.
- 635 5. Construct the final encoding by concatenating the following bit fields, from most
636 significant to least significant: Header 10 (2 bits), Filter Value F (3 bits), Company
637 Prefix Index C from Step 2 (14 bits), Item Reference from Step 3 (20 bits), Serial
638 Number S from Step 4 (25 bits).

639 **3.4.1.2 SGTIN-64 Decoding Procedure**

640 Given:

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

643 Yields:

644 An EAN.UCC GTIN-14

645 A Serial Number

646 A Filter Value

647 Procedure:

648 1. Bits $b_{61}b_{60}b_{59}$, considered as an unsigned integer, are the Filter Value.

649 2. Extract the Company Prefix Index C by considering bits $b_{58}b_{57}\dots b_{45}$ as an unsigned

650 integer.

651 3. Look up the Company Prefix Index C in the Company Prefix Translation Table to

652 obtain the EAN.UCC Company Prefix $p_1p_2\dots p_L$ consisting of L decimal digits (the value

653 of L is also obtained from the table). If the Company Prefix Index C is not found in the

654 Company Prefix Translation Table, stop: this bit string cannot be decoded as an SGTIN-

655 64.

656 4. Consider bits $b_{44}b_{43}\dots b_{25}$ as an unsigned integer. If this integer is greater than or

657 equal to $10^{(13-L)}$, stop: the input bit string is not a legal SGTIN-64 encoding. Otherwise,

658 convert this integer to a (13- L)-digit decimal number $i_1i_2\dots i_{(13-L)}$, adding leading zeros as

659 necessary to make (13- L) digits.

660 5. Construct a 13-digit number $d_1d_2\dots d_{13}$ where $d_1 = i_1$ from Step 4, $d_2d_3\dots d_{(L+1)} =$

661 $p_1p_2\dots p_L$ from Step 3, and $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2i_3\dots i_{(13-L)}$ from Step 4.

662 6. Calculate the check digit $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 +$

663 $d_8 + d_{10} + d_{12})) \bmod 10$.

664 7. The EAN.UCC GTIN-14 is the concatenation of digits from Steps 5 and 6: $d_1d_2\dots d_{14}$.

665 8. Bits $b_{24}b_{23}\dots b_0$, considered as an unsigned integer, are the Serial Number.

666 9. (Optional) If it is desired to represent the serial number as a UCC/EAN-128

667 Application Identifier 21, convert the integer from Step 8 to a decimal string with no

668 leading zeros. If the integer in Step 8 is zero, convert it to a string consisting of the single

669 character "0".

670 3.4.2 SGTIN-96

671 In addition to a Header, the SGTIN-96 is composed of five fields: the *Filter Value*,

672 *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)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,999,999 (Decimal capacity*)	9,999,999 – 9 (Decimal capacity*)	274,877,906,943 (Decimal capacity)

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

674 **Table 6.** The EPC SGTIN-96 bit allocation, header, and decimal capacity.

675 *Header* is 8-bits, with a binary value of 0011 0000.

676 *Filter Value* is not part of the GTIN or EPC identifier, but is used for fast filtering and
677 pre-selection of basic logistics types, such as items, inner packs, cases and pallets. The
678 Filter Values for 64-bit and 96-bit GTIN are the same. See Table 5.

679 *Partition* is an indication of where the subsequent Company Prefix and Item Reference
680 numbers are divided. This organization matches the structure in the EAN.UCC GTIN in
681 which the Company Prefix added to the Item Reference number (plus the single Indicator
682 Digit) totals 13 digits, yet the Company Prefix may vary from 6 to 12 digits and the Item
683 Reference (including the single Indicator Digit) from 7 to 1 digit(s). The available values
684 of *Partition* and the corresponding sizes of the *Company Prefix* and *Item Reference* fields
685 are defined in Table 7.

686 *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.

687 *Item Reference* contains a literal embedding of the GTIN Item Reference number. The
688 Indicator Digit is combined with the Item Reference field in the following manner:
689 Leading zeros on the item reference are significant. Put the Indicator Digit in the leftmost
690 position available within the field. *For instance, 00235 is different than 235. With the*
691 *indicator digit of 1, the combination with 00235 is 100235.* The resulting combination is
692 treated as a single integer, and encoded into binary to form the *Item Reference* field.

693 *Serial Number* contains a serial number. The SGTIN-96 encoding is only capable of
694 representing integer-valued serial numbers with limited range. Other EAN.UCC
695 specifications permit a broader range of serial numbers. In particular, the EAN-128
696 barcode symbology provides for a 20-character alphanumeric serial number to be
697 associated with a GTIN using Application Identifier (AI) 21 [EANUCCGS]. It is
698 possible to convert between the serial numbers in the SGTIN-96 tag encoding and the
699 serial numbers in AI 21 barcodes under certain conditions. Specifically, such
700 interconversion is possible when the alphanumeric serial number in AI 21 happens to
701 consist only of digit characters, with no leading zeros, and whose value when interpreted
702 as an integer falls within the range limitations of the SGTIN-96 tag encoding. These
703 considerations are reflected in the encoding and decoding procedures below.

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

705

Table 7. SGTIN-96 Partitions.

706 3.4.2.1 SGTIN-96 Encoding Procedure

707 The following procedure creates an SGTIN-96 encoding.

708 Given:

709 An EAN.UCC GTIN-14 consisting of digits $d_1d_2\dots d_{14}$

710 The length L of the Company Prefix portion of the GTIN

711 A Serial Number S where $0 \leq S < 2^{38}$, or an UCC/EAN-128 Application Identifier 21
712 consisting of characters $s_1s_2\dots s_K$.

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

714 Procedure:

715 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column
716 of the Partition Table (Table 7) to determine the Partition Value, P , the number of bits M
717 in the Company Prefix field, and the number of bits N in the Item Reference and
718 Indicator Digit field. If L is not found in any row of Table 7, stop: this GTIN cannot be
719 encoded in an SGTIN-96.

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

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

724 4. When the Serial Number is provided directly as an integer S where $0 \leq S < 2^{38}$,
725 proceed to Step 5. Otherwise, when the Serial Number is provided as an UCC/EAN-128
726 Application Identifier 21 consisting of characters $s_1s_2\dots s_K$, construct the Serial Number
727 by concatenating digits $s_1s_2\dots s_K$. If any of these characters is not a digit, stop: this Serial

728 Number cannot be encoded in the SGTIN-96 encoding. Also, if $K > 1$ and $s_1 = 0$, stop:
 729 this Serial Number cannot be encoded in the SGTIN-96 encoding (because leading zeros
 730 are not permitted except in the case where the Serial Number consists of a single zero
 731 digit). Otherwise, consider the result to be a decimal integer, S . If $S \geq 2^{38}$, stop: this
 732 Serial Number cannot be encoded in the SGTIN-96 encoding.

733 5. Construct the final encoding by concatenating the following bit fields, from most
 734 significant to least significant: Header 00110000 (8 bits), Filter Value F (3 bits),
 735 Partition Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Item
 736 Reference from Step 3 (N bits), Serial Number S from Step 4 (38 bits). Note that $M+N =$
 737 44 bits for all P .

738 3.4.2.2 SGTIN-96 Decoding Procedure

739 Given:

740 An SGTIN-96 as a 96-bit bit string $00110000b_{87}b_{86}\dots b_0$ (where the first eight bits
 741 00110000 are the header)

742 Yields:

743 An EAN.UCC GTIN-14

744 A Serial Number

745 A Filter Value

746 Procedure:

- 747 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.
- 748 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
 749 $P = 7$, stop: this bit string cannot be decoded as an SGTIN-96.
- 750 3. Look up the Partition Value P in Table 7 to obtain the number of bits M in the
 751 Company Prefix and the number of digits L in the Company Prefix.
- 752 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned
 753 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a
 754 legal SGTIN-96 encoding. Otherwise, convert this integer into a decimal number
 755 $p_1p_2\dots p_L$, adding leading zeros as necessary to make up L digits in total.
- 756 5. Extract the Item Reference and Indicator by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{38}$ as an
 757 unsigned integer. If this integer is greater than or equal to $10^{(13-L)}$, stop: the input bit
 758 string is not a legal SGTIN-96 encoding. Otherwise, convert this integer to a (13- L)-digit
 759 decimal number $i_1i_2\dots i_{(13-L)}$, adding leading zeros as necessary to make (13- L) digits.
- 760 6. Construct a 13-digit number $d_1d_2\dots d_{13}$ where $d_1 = i_1$ from Step 5, $d_2d_3\dots d_{(L+1)} =$
 761 $p_1p_2\dots p_L$ from Step 4, and $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2i_3\dots i_{(13-L)}$ from Step 5.
- 762 7. Calculate the check digit $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 +$
 763 $d_8 + d_{10} + d_{12})) \bmod 10$.
- 764 8. The EAN.UCC GTIN-14 is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{14}$.
- 765 9. Bits $b_{37}b_{36}\dots b_0$, considered as an unsigned integer, are the Serial Number.

766 10. (Optional) If it is desired to represent the serial number as a UCC/EAN-128
 767 Application Identifier 21, convert the integer from Step 9 to a decimal string with no
 768 leading zeros. If the integer in Step 9 is zero, convert it to a string consisting of the single
 769 character “0”.

770 3.5 Serial Shipping Container Code (SSCC)

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

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

783 3.5.1 SSCC-64

784 In addition to a Header, the EPC SSCC-64 is composed of three fields: the *Filter Value*,
 785 *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)	8 (Decimal capacity)	16,383 (Decimal capacity)	99,999 - 99,999,999,999 (Decimal capacity*)

786 *Capacity of Serial Reference field varies with the length of the Company Prefix

787 **Table 8.** The EPC 64-bit SSCC bit allocation, header, and decimal capacity.

788

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

790 *Filter Value* is not part of the SSCC or EPC identifier, but is used for fast filtering and
 791 pre-selection of basic logistics types, such as cases and pallets. The Filter Values for 64-
 792 bit and 96-bit SSCC are the same. The normative specifications for Filter Values are
 793 specified in Table 9. The value of 000 means “unspecified”. That is, a filter value of 000
 794 means that the tag provides no information as to the logistics type of the object to which
 795 the tag is affixed. In particular, tags conforming to earlier versions of this specification,

796 in which 000 was the only value approved for use, will have filter equal to 000. In
 797 general, filter 000 should not be used for SSCC EPC tags intended for use in the supply
 798 chain following ratification of this standard.

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

799 **Table 9.** SSCC Filter Values

800 *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this field is
 801 not the Company Prefix itself, but rather an index into a table that provides the Company
 802 Prefix as well as an indication of the Company Prefix's length. The means by which
 803 hardware or software may obtain the contents of the translation table is specified in
 804 [Translation of 64-bit Tag Encoding Company Prefix Indices Into EAN.UCC Company
 805 Prefixes].

806 *Serial Reference* is a unique number for each instance, comprised of the Serial Reference
 807 and the Extension digit. The Extension Digit is combined with the Serial Reference field
 808 in the following manner: Leading zeros on the Serial Reference are significant. Put the
 809 Extension Digit in the leftmost position available within the field. *For instance,*
 810 *000042235 is different than 42235. With the extension digit of 1, the combination with*
 811 *000042235 is 1000042235.* The resulting combination is treated as a single integer, and
 812 encoded into binary to form the Serial Reference field. To avoid unmanageably large and
 813 out-of-specification serial references, they should not exceed the capacity specified in
 814 EAN.UCC specifications, which are (inclusive of extension digit) 9,999 for company
 815 prefixes of 12 digits up to 9,999,999,999 for company prefixes of 6 digits.

Formatted: Indent: Left: 0 pt, First line: 0 pt, Bulleted + Level: 1 + Aligned at: 18 pt + Tab after: 36 pt + Indent at: 36 pt, Tabs: 0 pt, List tab + Not at 36 pt

816 **3.5.1.1 SSCC-64 Encoding Procedure**

817 The following procedure creates an SSCC-64 encoding.

818 Given:

819 An EAN.UCC SSCC consisting of digits $d_1d_2\dots d_{18}$

820 The length L of the company prefix portion of the SSCC

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

822 Procedure:

- 823 1. Extract the EAN.UCC Company Prefix $d_2d_3\dots d_{(L+1)}$
- 824 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table
825 to obtain the corresponding Company Prefix Index, C . If the Company Prefix was not
826 found in the Company Prefix Translation Table, stop: this SSCC cannot be encoded in
827 the SSCC-64 encoding.
- 828 3. Construct the Serial Reference by concatenating digits $d_1d_{(L+2)}d_{(L+3)}\dots d_{17}$ and
829 considering the result to be a decimal integer, I . If $I \geq 2^{39}$, stop: this SSCC cannot be
830 encoded in the SSCC-64 encoding.
- 831 4. Construct the final encoding by concatenating the following bit fields, from most
832 significant to least significant: Header 00001000 (8 bits), Filter Value F (3 bits),
833 Company Prefix Index C from Step 2 (14 bits), Serial Reference from Step 3 (39 bits).

834 3.5.1.2 SSCC-64 Decoding Procedure

835 Given:

836 An SSCC-64 as a 64-bit bit string $00001000b_{55}b_{54}\dots b_0$ (where the first eight bits
837 00001000 are the header)

838 Yields:

839 An EAN.UCC SSCC

840 A Filter Value

841 Procedure:

- 842 1. Bits $b_{55}b_{54}b_{53}$, considered as an unsigned integer, are the Filter Value.
- 843 2. Extract the Company Prefix Index C by considering bits $b_{52}b_{51}\dots b_{39}$ as an unsigned
844 integer.
- 845 3. Look up the Company Prefix Index C in the Company Prefix Translation Table to
846 obtain the EAN.UCC Company Prefix $p_1p_2\dots p_L$ consisting of L decimal digits (the value
847 of L is also obtained from the table). If the Company Prefix Index C is not found in the
848 Company Prefix Translation Table, stop: this bit string cannot be decoded as an SSCC-
849 64.
- 850 4. Consider bits $b_{38}b_{37}\dots b_0$ as an unsigned integer. If this integer is greater than or equal
851 to $10^{(17-L)}$, stop: the input bit string is not a legal SSCC-64 encoding. Otherwise, convert
852 this integer to a $(17-L)$ -digit decimal number $i_1i_2\dots i_{(17-L)}$, adding leading zeros as
853 necessary to make $(17-L)$ digits.
- 854 5. Construct a 17-digit number $d_1d_2\dots d_{17}$ where $d_1 = s_1$ from Step 4, $d_2d_3\dots d_{(L+1)} =$
855 $p_1p_2\dots p_L$ from Step 3, and $d_{(L+2)}d_{(L+3)}\dots d_{17} = i_2i_3\dots i_{(17-L)}$ from Step 4.
- 856 6. Calculate the check digit $d_{18} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13} + d_{15} + d_{17}) - (d_2 +$
857 $d_4 + d_6 + d_8 + d_{10} + d_{12} + d_{14} + d_{16})) \bmod 10$.
- 858 7. The EAN.UCC SSCC is the concatenation of digits from Steps 5 and 6: $d_1d_2\dots d_{18}$.

859 **3.5.2 SSCC-96**

860 In addition to a Header, the EPC SSCC-96 is composed of four fields: the *Filter Value*,
 861 *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)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,999,999 (Decimal capacity*)	99,999,999,999 – 99,999 (Decimal capacity*)	[Not Used]

862 *Capacity of Company Prefix and Serial Reference fields vary according to the contents of the Partition field.

863 **Table 10.** The EPC 96-bit SSCC bit allocation, header, and decimal capacity.

864 *Header* is 8-bits, with a binary value of 0011 0001.

865 *Filter Value* is not part of the SSCC or EPC identifier, but is used for fast filtering and
 866 pre-selection of basic logistics types, such as cases and pallets. The Filter Values for 64-
 867 bit and 96-bit SSCC are the same. See Table 9.

868 The *Partition* is an indication of where the subsequent Company Prefix and Serial
 869 Reference numbers are divided. This organization matches the structure in the
 870 EAN.UCC SSCC in which the Company Prefix added to the Serial Reference number
 871 (including the single Extension Digit) totals 17 digits, yet the Company Prefix may vary
 872 from 6 to 12 digits and the Serial Reference from 11 to 5 digit(s). Table 11 shows
 873 allowed values of the partition value and the corresponding lengths of the company prefix
 874 and serial reference.

Partition Value (P)	Company Prefix		Serial Reference and Extension Digit	
	Bits (M)	Digits (L)	Bits (N)	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

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
6	20	6	38	11

Table 11. SSCC-96 Partitions.

875

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

877 *Serial Reference* is a unique number for each instance, comprised of the Serial Reference
878 and the Extension digit. The Extension Digit is combined with the Serial Reference field
879 in the following manner: Leading zeros on the Serial Reference are significant. Put the
880 Extension Digit in the leftmost position available within the field. *For instance,*
881 *000042235 is different than 42235. With the extension digit of 1, the combination with*
882 *000042235 is 1000042235.* The resulting combination is treated as a single integer, and
883 encoded into binary to form the Serial Reference field. To avoid unmanageably large and
884 out-of-specification serial references, they should not exceed the capacity specified in
885 EAN.UCC specifications, which are (inclusive of extension digit) 9,999 for company
886 prefixes of 12 digits up to 9,999,999,999 for company prefixes of 6 digits.

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

889 **3.5.2.1 SSCC-96 Encoding Procedure**

890 The following procedure creates an SSCC-96 encoding.

891 Given:

892 An EAN.UCC SSCC consisting of digits $d_1d_2\dots d_{18}$

893 The length L of the Company Prefix portion of the SSCC

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

895 Procedure:

896 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column
897 of the Partition Table (Table 11) to determine the Partition Value, P , the number of bits
898 M in the Company Prefix field, and the number of bits N in the Serial Reference and
899 Extension Digit field. If L is not found in any row of Table 11, stop: this SSCC cannot
900 be encoded in an SSCC-96.

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

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

905 4. Construct the final encoding by concatenating the following bit fields, from most
906 significant to least significant: Header 00110001 (8 bits), Filter Value F (3 bits),

907 Partition Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Serial
908 Reference S from Step 3 (N bits), and 24 zero bits. Note that $M+N = 58$ bits for all P .

909 **3.5.2.2 SSCC-96 Decoding Procedure**

910 Given:

911 An SSCC-96 as a 96-bit bit string $00110001b_{87}b_{86}\dots b_0$ (where the first eight bits
912 00110001 are the header)

913 Yields:

914 An EAN.UCC SSCC

915 A Filter Value

916 Procedure:

- 917 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.
- 918 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
919 $P = 7$, stop: this bit string cannot be decoded as an SSCC-96.
- 920 3. Look up the Partition Value P in Table 11 to obtain the number of bits M in the
921 Company Prefix and the number of digits L in the Company Prefix.
- 922 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned
923 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a
924 legal SSCC-96 encoding. Otherwise, convert this integer into a decimal number
925 $p_1p_2\dots p_L$, adding leading zeros as necessary to make up L digits in total.
- 926 5. Extract the Serial Reference by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{24}$ as an unsigned
927 integer. If this integer is greater than or equal to $10^{(17-L)}$, stop: the input bit string is not a
928 legal SSCC-96 encoding. Otherwise, convert this integer to a $(17-L)$ -digit decimal
929 number $i_1i_2\dots i_{(17-L)}$, adding leading zeros as necessary to make $(17-L)$ digits.
- 930 6. Construct a 17-digit number $d_1d_2\dots d_{17}$ where $d_1 = s_1$ from Step 5, $d_2d_3\dots d_{(L+1)} =$
931 $p_1p_2\dots p_L$ from Step 4, and $d_{(L+2)}d_{(L+3)}\dots d_{17} = i_2i_3\dots i_{(17-L)}$ from Step 5.
- 932 7. Calculate the check digit $d_{18} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13} + d_{15} + d_{17}) - (d_2 +$
933 $d_4 + d_6 + d_8 + d_{10} + d_{12} + d_{14} + d_{16})) \bmod 10$.
- 934 8. The EAN.UCC SSCC is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{18}$.

935 **3.6 Serialized Global Location Number (SGLN)**

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

940 In the SGLN-64 encoding, the limited number of bits prohibits a literal embedding of the
941 GLN. As a partial solution, a Company Prefix *Index* is used. This *index*, which can
942 accommodate up to 16,384 codes, is assigned to companies that need to use the 64 bit
943 tags, in addition to their existing EAN.UCC Company Prefixes. The *index* is encoded on

944 the tag instead of the Company Prefix, and is subsequently translated to the Company
 945 Prefix at low levels of the EPC system components (i.e. the Reader or Savant).

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

949 **3.6.1 SGLN-64**

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

952

	Header	Filter Value	Company Prefix Index	Location Reference	Serial Number
SGLN-64	8	3	14	20	19
	0000 1001 (Binary value)	8 (Decimal capacity)	16,383 (Decimal capacity)	999,999 - 0 (Decimal capacity*)	524,288 (Decimal capacity) [Not Used]

953 *Capacity of Location Reference field varies with the length of the Company Prefix

954 **Table 12.** The EPC SGLN-64 bit allocation, header, and decimal capacity.

955

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

957 *Filter Value* is not part of the SGLN pure identity, but is additional data that is used for
 958 fast filtering and pre-selection of basic location types. The Filter Values for 64-bit and
 959 96-bit SGLN are the same. See Table 13 for currently defined filter values.

960 *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this field is
 961 not the Company Prefix itself, but rather an index into a table that provides the Company
 962 Prefix as well as an indication of the Company Prefix's length. The means by which
 963 hardware or software may obtain the contents of the translation table is specified in
 964 [Translation of 64-bit Tag Encoding Company Prefix Indices Into EAN.UCC Company
 965 Prefixes].

966 *Location Reference* encodes the GLN Location Reference number.

967 *Serial Number* contains a serial number. Note: The serial number field is reserved and
 968 should not be used, until the EAN.UCC community determines the appropriate way, if
 969 any, for extending GLN.

970

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

971

972

Table 13. SGLN Filter Values .

973 **3.6.1.1 SGLN-64 Encoding Procedure**

974 The following procedure creates an SGLN-64 encoding.

975 Given:

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

977 The length L of the company prefix portion of the GLN

978 A Serial Number S where $0 \leq S < 2^{19}$

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

980 Procedure:

981 1. Extract the EAN.UCC Company Prefix $d_1d_2\dots d_L$

982 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table
 983 to obtain the corresponding Company Prefix Index, C . If the Company Prefix was not
 984 found in the Company Prefix Translation Table, stop: this GLN cannot be encoded in the
 985 SGLN-64 encoding.

986 3. Construct the Location Reference by concatenating digits $d_{(L+1)}d_{(L+2)}\dots d_{12}$ and
 987 considering the result to be a decimal integer, I . If $I \geq 2^{20}$, stop: this GLN cannot be
 988 encoded in the SGLN-64 encoding.

989 4. Construct the final encoding by concatenating the following bit fields, from most
 990 significant to least significant: Header 00001001 (8 bits), Filter Value F (3 bits),
 991 Company Prefix Index C from Step 2 (14 bits), Location Reference from Step 3 (20 bits),
 992 Serial Number S (19 bits).

993 **3.6.1.2 SGLN-64 Decoding Procedure**

994 Given:

995 An SGLN-64 as a 64-bit bit string $00001001b_{55}b_{54}\dots b_0$ (where the first eight bits
996 00001001 are the header)
997 Yields:
998 An EAN.UCC GLN
999 A Serial Number
1000 A Filter Value
1001 Procedure:
1002 1. Bits $b_{55}b_{54}b_{53}$, considered as an unsigned integer, are the Filter Value.
1003 2. Extract the Company Prefix Index C by considering bits $b_{52}b_{51}\dots b_{39}$ as an unsigned
1004 integer.
1005 3. Look up the Company Prefix Index C in the Company Prefix Translation Table to
1006 obtain the EAN.UCC Company Prefix $p_1p_2\dots p_L$ consisting of L decimal digits (the value
1007 of L is also obtained from the table). If the Company Prefix Index C is not found in the
1008 Company Prefix Translation Table, stop: this bit string cannot be decoded as an SGLN-
1009 64.
1010 4. Consider bits $b_{38}b_{37}\dots b_{19}$ as an unsigned integer. If this integer is greater than or
1011 equal to $10^{(12-L)}$, stop: the input bit string is not a legal SGLN-64 encoding. Otherwise,
1012 convert this integer to a $(12-L)$ -digit decimal number $i_1i_2\dots i_{(12-L)}$, adding leading zeros as
1013 necessary to make $(12-L)$ digits.
1014 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
1015 $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_1i_2\dots i_{(12-L)}$ from Step 4.
1016 6. Calculate the check digit $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 +$
1017 $d_9 + d_{11})) \bmod 10$.
1018 7. The EAN.UCC GLN is the concatenation of digits from Steps 5 and 6: $d_1d_2\dots d_{13}$.
1019 8. Bits $b_{18}b_{17}\dots b_0$, considered as an unsigned integer, are the Serial Number.

1020 **3.6.2 SGLN-96**

1021 In addition to a Header, the SGLN-96 is composed of five fields: the *Filter Value*,
1022 *Partition*, *Company Prefix*, *Location Reference*, and *Serial Number*, as shown in Table 14.
1023 *Header* is 8-bits, with a binary value of 0011 0010.
1024 *Filter Value* is not part of the GLN or EPC identifier, but is used for fast filtering and pre-
1025 selection of basic location types. The Filter Values for 64-bit and 96-bit GLN are the
1026 same. See Table 13.
1027 *Partition* is an indication of where the subsequent Company Prefix and Location
1028 Reference numbers are divided. This organization matches the structure in the
1029 EAN.UCC GLN in which the Company Prefix added to the Location Reference number
1030 totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the Location
1031 Reference number from 6 to 0 digit(s). The available values of *Partition* and the

1032 corresponding sizes of the *Company Prefix* and *Location Reference* fields are defined in
 1033 Table 15.

1034

	Header	Filter Value	Partition	Company Prefix	Location Reference	Serial Number
SGLN-96	8	3	3	20-40	21-1	41
	0011 0010 (Binary value)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,99 9,999 (Decimal capacity*)	999,999 – 0 (Decimal capacity*)	2,199,023,255 ,552 (Decimal capacity) [Not Used]

1035 *Capacity of Company Prefix and Location Reference fields vary according to contents of the Partition field.

1036 **Table 14.** The EPC SGLN-96 bit allocation, header, and decimal capacity.

1037

1038 *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.

1039 *Location Reference* encodes the GLN Location Reference number.

1040 *Serial Number* contains a serial number. Note: The serial number field is reserved and
 1041 should not be used, until the EAN.UCC community determines the appropriate way, if
 1042 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

1043

Table 15. SGLN-96 Partitions.

1044 **3.6.2.1 SGLN-96 Encoding Procedure**

1045 The following procedure creates an SGLN-96 encoding.

1046 Given:

1047 An EAN.UCC GLN consisting of digits $d_1d_2\dots d_{13}$

1048 The length L of the Company Prefix portion of the GLN

1049 A Serial Number S where $0 \leq S < 2^{41}$

1050 A Filter Value F where $0 \leq F < 8$

1051 Procedure:

1052 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column
1053 of the Partition Table (Table 15) to determine the Partition Value, P , the number of bits
1054 M in the Company Prefix field, and the number of bits N in the Location Reference field.
1055 If L is not found in any row of Table 15, stop: this GLN cannot be encoded in an SGLN-
1056 96.

1057 2. Construct the Company Prefix by concatenating digits $d_1d_2\dots d_L$ and considering the
1058 result to be a decimal integer, C .

1059 3. Construct the Location Reference by concatenating digits $d_{(L+1)}d_{(L+2)}\dots d_{12}$ and
1060 considering the result to be a decimal integer, I .

1061 4. Construct the final encoding by concatenating the following bit fields, from most
1062 significant to least significant: Header 00110010 (8 bits), Filter Value F (3 bits),
1063 Partition Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Location
1064 Reference from Step 3 (N bits), Serial Number S (41 bits). Note that $M+N = 41$ bits for
1065 all P .

1066 **3.6.2.2 SGLN-96 Decoding Procedure**

1067 Given:

1068 An SGLN-96 as a 96-bit bit string $00110010b_{87}b_{86}\dots b_0$ (where the first eight bits
1069 00110010 are the header)

1070

1071 Yields:

1072 An EAN.UCC GLN

1073 A Serial Number

1074 A Filter Value

1075 Procedure:

1076 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.

1077 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
1078 $P = 7$, stop: this bit string cannot be decoded as an SGLN-96.

- 1079 3. Look up the Partition Value P in Table 15 to obtain the number of bits M in the
1080 Company Prefix and the number of digits L in the Company Prefix.
- 1081 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned
1082 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a
1083 legal SGLN-96 encoding. Otherwise, convert this integer into a decimal number
1084 $p_1p_2\dots p_L$, adding leading zeros as necessary to make up L digits in total.
- 1085 5. Extract the Location Reference by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{41}$ as an unsigned
1086 integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a
1087 legal SGLN-96 encoding. Otherwise, convert this integer to a $(12-L)$ -digit decimal
1088 number $i_1i_2\dots i_{(12-L)}$, adding leading zeros as necessary to make $(12-L)$ digits.
- 1089 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
1090 $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_2i_3\dots i_{(12-L)}$ from Step 5.
- 1091 7. Calculate the check digit $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 +$
1092 $d_9 + d_{11})) \bmod 10$.
- 1093 8. The EAN.UCC GLN is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{13}$.
- 1094 9. Bits $b_{40}b_{39}\dots b_0$, considered as an unsigned integer, are the Serial Number.

1095 **3.7 Global Returnable Asset Identifier (GRAI)**

1096 The EPC encoding scheme for GRAI permits the direct embedding of EAN.UCC System
1097 standard GRAI on EPC tags. In all cases, the check digit is not encoded. Two encoding
1098 schemes are specified, GRAI-64 (64 bits) and GRAI-96 (96 bits).

1099 In the GRAI-64 encoding, the limited number of bits prohibits a literal embedding of the
1100 GRAI. As a partial solution, a Company Prefix *Index* is used. This Index, which can
1101 accommodate up to 16,384 codes, is assigned to companies that need to use the 64 bit
1102 tags, in addition to their existing EAN.UCC Company Prefixes. The Index is encoded on
1103 the tag instead of the Company Prefix, and is subsequently translated to the Company
1104 Prefix at low levels of the EPC system components (i.e. the Reader or Savant). While
1105 this means that only a limited number of Company Prefixes can be represented in the 64-
1106 bit tag, this is a transitional step to full accommodation in 96-bit and additional encoding
1107 schemes.

1108 **3.7.1 GRAI-64**

1109 The GRAI-64 includes *four* fields in addition to the Header – *Filter Value*, *Company*
1110 *Prefix Index*, *Asset Type*, and *Serial Number*, as shown in Table 16.

1111

1112

1113

1114

1115

1116
 1117
 1118

	Header	Filter Value	Company Prefix Index	Asset Type	Serial Number
GRAI-64	8	3	14	20	19
	0000 1010 (Binary value)	8 (Decimal capacity)	16,383 (Decimal capacity)	999,999 - 0 (Decimal capacity*)	524,288 (Decimal capacity)

1119 *Capacity of Asset Type field varies with Company Prefix.

1120 **Table 16.** The EPC GRAI-64 bit allocation, header, and decimal capacity.

1121

1122 *Header* is 8 bits, with a binary value of 0000 1010.

1123 *Filter Value* is not part of the GRAI pure identity, but is additional data that is used for
 1124 fast filtering and pre-selection of basic asset types. The Filter Values for 64-bit and 96-
 1125 bit GRAI are the same. See Table 17 for currently defined GRAI filter values. This
 1126 specification anticipates that valuable Filter Values will be determined once there has
 1127 been time to consider the possible use cases.

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

1128

1129 **Table 17.** GRAI Filter Values

1130 *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this field is
 1131 not the Company Prefix itself, but rather an index into a table that provides the Company

1132 Prefix as well as an indication of the Company Prefix's length. The means by which
1133 hardware or software may obtain the contents of the translation table is specified in
1134 [Translation of 64-bit Tag Encoding Company Prefix Indices Into EAN.UCC Company
1135 Prefixes].

1136 *Asset Type* encodes the GRAI Asset Type number.

1137 *Serial Number* contains a serial number. The 64-bit and 96-bit tag encodings are only
1138 capable of representing a subset of Serial Numbers allowed in the General EAN.UCC
1139 Specifications. The capacity of this mandatory serial number is less than the maximum
1140 EAN.UCC System specification for serial number, no leading zeros are permitted, and
1141 only numbers are permitted.

1142 **3.7.1.1 GRAI-64 Encoding Procedure**

1143 The following procedure creates a GRAI-64 encoding.

1144 Given:

1145 An EAN.UCC GRAI consisting of digits $0d_2\dots d_K$, where $15 \leq K \leq 30$.

1146 The length L of the company prefix portion of the GRAI

1147 A Filter Value F where $0 \leq F < 8$

1148 Procedure:

1149 1. Extract the EAN.UCC Company Prefix $d_2d_3\dots d_{L+1}$

1150 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table
1151 to obtain the corresponding Company Prefix Index, C . If the Company Prefix was not
1152 found in the Company Prefix Translation Table, stop: this GRAI cannot be encoded in
1153 the GRAI-64 encoding.

1154 3. Construct the Asset Type by concatenating digits $d_{(L+2)}d_{(L+3)}\dots d_{13}$ and considering the
1155 result to be a decimal integer, I . If $I \geq 2^{20}$, stop: this GRAI cannot be encoded in the
1156 GRAI-64 encoding.

1157 4. Construct the Serial Number by concatenating digits $d_{15}d_{16}\dots d_K$. If any of these
1158 characters is not a digit, stop: this GRAI cannot be encoded in the GRAI-64 encoding.
1159 Otherwise, consider the result to be a decimal integer, S . If $S \geq 2^{19}$, stop: this GRAI
1160 cannot be encoded in the GRAI-64 encoding. Also, if $K > 15$ and $d_{15} = 0$, stop: this
1161 GRAI cannot be encoded in the GRAI-64 encoding (because leading zeros are not
1162 permitted except in the case where the Serial Number consists of a single zero digit).

1163 5. Construct the final encoding by concatenating the following bit fields, from most
1164 significant to least significant: Header 00001010 (8 bits), Filter Value F (3 bits),
1165 Company Prefix Index C from Step 2 (14 bits), Asset Type I from Step 3 (20 bits), Serial
1166 Number S from Step 4 (19 bits).

1167 **3.7.1.2 GRAI-64 Decoding Procedure**

1168 Given:

1169 An GRAI-64 as a 64-bit bit string $00001010b_{55}b_{54}\dots b_0$ (where the first eight bits
1170 00001010 are the header)
1171 Yields:
1172 An EAN.UCC GRAI
1173 A Filter Value
1174 Procedure:
1175 1. Bits $b_{55}b_{54}b_{53}$, considered as an unsigned integer, are the Filter Value.
1176 2. Extract the Company Prefix Index C by considering bits $b_{52}b_{51}\dots b_{39}$ as an unsigned
1177 integer.
1178 3. Look up the Company Prefix Index C in the Company Prefix Translation Table to
1179 obtain the EAN.UCC Company Prefix $p_1p_2\dots p_L$ consisting of L decimal digits (the value
1180 of L is also obtained from the table). If the Company Prefix Index C is not found in the
1181 Company Prefix Translation Table, stop: this bit string cannot be decoded as a GRAI-64.
1182 4. Consider bits $b_{38}b_{37}\dots b_{19}$ as an unsigned integer. If this integer is greater than or
1183 equal to $10^{(12-L)}$, stop: the input bit string is not a legal GRAI-64 encoding. Otherwise,
1184 convert this integer to a $(12-L)$ -digit decimal number $i_1i_2\dots i_{(12-L)}$, adding leading zeros as
1185 necessary to make $(12-L)$ digits.
1186 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
1187 $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_1i_2\dots i_{(12-L)}$ from Step 4.
1188 6. Calculate the check digit $d_{14} = (-3(d_3 + d_5 + d_7 + d_9 + d_{11}+d_{13}) - (d_2 + d_4 + d_6 + d_8 +$
1189 $d_{10} + d_{12})) \bmod 10$.
1190 7. Consider bits $b_{18}b_{17}\dots b_0$ as an unsigned integer. Convert this integer into a decimal
1191 number $d_{15}d_{16}\dots d_K$, with no leading zeros (exception: if the integer is equal to zero,
1192 convert it to a single zero digit).
1193 8. The EAN.UCC GRAI is the concatenation of the digits from Steps 5, 6, and 7:
1194 $0d_2d_3\dots d_K$.

1195 3.7.2 GRAI-96

1196 In addition to a Header, the GRAI-96 is composed of five fields: the *Filter Value*,
1197 *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)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,9 99,999 (Decimal capacity*)	999,999 – 0 (Decimal capacity*)	274,877,906 ,943 (Decimal capacity)

1198 *Capacity of Company Prefix and Asset Type fields vary according to contents of the Partition field.

1199 **Table 18.** The EPC GRAI-96 bit allocation, header, and decimal capacity.

1200 *Header* is 8-bits, with a binary value of 0011 0011.

1201 *Filter Value* is not part of the GRAI or EPC identifier, but is used for fast filtering and
1202 pre-selection of basic asset types. The Filter Values for 64-bit and 96-bit GRAI are the
1203 same. See Table 17.

1204 *Partition* is an indication of where the subsequent Company Prefix and Asset Type
1205 numbers are divided. This organization matches the structure in the EAN.UCC GRAI in
1206 which the Company Prefix added to the Asset Type number totals 12 digits, yet the
1207 Company Prefix may vary from 6 to 12 digits and the Asset Type from 6 to 0 digit(s).
1208 The available values of *Partition* and the corresponding sizes of the *Company Prefix* and
1209 *Asset Type* fields are defined in Table 19.

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

1210 **Table 19.** GRAI-96 Partitions.

1211

1212 *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.

1213 *Asset Type* encodes the GRAI Asset Type number.

1214 *Serial Number* contains a serial number. The 64-bit and 96-bit tag encodings are only
1215 capable of representing a subset of Serial Numbers allowed in the General EAN.UCC
1216 Specifications. The capacity of this mandatory serial number is less than the maximum
1217 EAN.UCC System specification for serial number, no leading zeros are permitted, and
1218 only numbers are permitted.

1219 3.7.2.1 GRAI-96 Encoding Procedure

1220 The following procedure creates a GRAI-96 encoding.

1221 Given:

1222 An EAN.UCC GRAI consisting of digits $0d_2d_3\dots d_K$, where $15 \leq K \leq 30$.

1223 The length L of the Company Prefix portion of the GRAI

1224 A Filter Value F where $0 \leq F < 8$

1225 Procedure:

1226 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column
1227 of the Partition Table (Table 19) to determine the Partition Value, P , the number of bits
1228 M in the Company Prefix field, and the number of bits N in Asset Type field. If L is not
1229 found in any row of Table 19, stop: this GRAI cannot be encoded in a GRAI-96.

1230 2. Construct the Company Prefix by concatenating digits $d_2d_3\dots d_{(L+1)}$ and considering
1231 the result to be a decimal integer, C .

1232 3. Construct the Asset Type by concatenating digits $d_{(L+2)}d_{(L+3)}\dots d_{13}$ and considering the
1233 result to be a decimal integer, I .

1234 4. Construct the Serial Number by concatenating digits $d_{15}d_{16}\dots d_K$. If any of these
1235 characters is not a digit, stop: this GRAI cannot be encoded in the GRAI-96 encoding.
1236 Otherwise, consider the result to be a decimal integer, S . If $S \geq 2^{38}$, stop: this GRAI
1237 cannot be encoded in the GRAI-96 encoding. Also, if $K > 15$ and $d_{15} = 0$, stop: this
1238 GRAI cannot be encoded in the GRAI-96 encoding (because leading zeros are not
1239 permitted except in the case where the Serial Number consists of a single zero digit).

1240 5. Construct the final encoding by concatenating the following bit fields, from most
1241 significant to least significant: Header 00110011 (8 bits), Filter Value F (3 bits),
1242 Partition Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Asset
1243 Type I from Step 3 (N bits), Serial Number S from Step 4 (38 bits). Note that $M+N =$
1244 44 bits for all P .

1245 **3.7.2.2 GRAI-96 Decoding Procedure**

1246 Given:

1247 An GRAI-96 as a 96-bit bit string 00110011 $b_{87}b_{86}\dots b_0$ (where the first eight bits
1248 00110011 are the header)

1249 Yields:

1250 An EAN.UCC GRAI

1251 A Filter Value

1252 Procedure:

1253 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.

1254 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
1255 $P = 7$, stop: this bit string cannot be decoded as a GRAI-96.

1256 3. Look up the Partition Value P in Table 19 to obtain the number of bits M in the
1257 Company Prefix and the number of digits L in the Company Prefix.

- 1258 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned
 1259 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a
 1260 legal GRAI-96 encoding. Otherwise, convert this integer into a decimal number
 1261 $p_1p_2\dots p_L$, adding leading zeros as necessary to make up L digits in total.
- 1262 5. Extract the Asset Type by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{38}$ as an unsigned integer.
 1263 If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a legal
 1264 GRAI-96 encoding. Otherwise, convert this integer to a $(12-L)$ -digit decimal number
 1265 $i_1i_2\dots i_{(12-L)}$, adding leading zeros as necessary to make $(12-L)$ digits.
- 1266 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,
 1267 and $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_1i_2\dots i_{(12-L)}$ from Step 5.
- 1268 7. Calculate the check digit $d_{14} = (-(-3(d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8$
 1269 $+ d_{10} + d_{12})) \bmod 10$.
- 1270 8. Extract the Serial Number by considering bits $b_{37}b_{36}\dots b_0$ as an unsigned integer.
 1271 Convert this integer to a decimal number $d_{15}d_{16}\dots d_K$, with no leading zeros (exception: if
 1272 the integer is equal to zero, convert it to a single zero digit).
- 1273 9. The EAN.UCC GRAI is the concatenation of a single zero digit and the digits from
 1274 Steps 6, 7, and 8: $0d_2d_3\dots d_K$.

1275 **3.8 Global Individual Asset Identifier (GIAI)**

1276 The EPC encoding scheme for GIAI permits the direct embedding of EAN.UCC System
 1277 standard GIAI codes on EPC tags (except as noted below for 64-bit tags). Two encoding
 1278 schemes are specified, GIAI-64 (64 bits) and GIAI-96 (96 bits).

1279 In the 64-bit EPC, the limited number of bits prohibits a literal embedding of the
 1280 EAN.UCC Company Prefix. As a partial solution, a Company Prefix *Index* is used. In
 1281 addition to their existing Company Prefixes, this Index, which can accommodate up to
 1282 16,384 codes, is assigned to companies that need to use the 64 bit tags. The Index is
 1283 encoded on the tag instead of the Company Prefix, and is subsequently translated to the
 1284 Company Prefix at low levels of the EPC system components (i.e. the Reader or Savant).
 1285 While this means a limited number of Company Prefixes can be represented in the 64-bit
 1286 tag, this is a transitional step to full accommodation in 96-bit and additional encoding
 1287 schemes.

1288 **3.8.1 GIAI-64**

1289 In addition to a Header, the EPC GIAI-64 is composed of three fields: the *Filter Value*,
 1290 *Company Prefix Index*, and *Individual Asset Reference*, as shown in Table 20.

1291

	Header	Filter Value	Company Prefix Index	Individual Asset Reference
GIAI-64	8	3	14	39

	0000 1011 (Binary value)	8 (Decimal capacity)	16,383 (Decimal capacity)	549,755,813,888 (Decimal capacity)
--	-----------------------------------	----------------------------	---------------------------------	--

1292 **Table 20.** The EPC 64-bit GIAI bit allocation, header, and decimal capacity.

1293

1294 *Header* is 8-bits, with a binary value of 0000 1011.

1295 *Filter Value* is not part of the GIAI pure identity, but is additional data that is used for
 1296 fast filtering and pre-selection of basic asset types. The Filter Values for 64-bit and 96-
 1297 bit GIAI are the same. See Table 21 for currently defined GIAI filter values. This
 1298 specification anticipates that valuable Filter Values will be determined once there has
 1299 been time to consider the possible use cases.

1300

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

1301 **Table 21.** GIAI Filter Values

1302 *Company Prefix Index* encodes the EAN.UCC Company Prefix. The value of this field is
 1303 not the Company Prefix itself, but rather an index into a table that provides the Company
 1304 Prefix as well as an indication of the Company Prefix's length. The means by which
 1305 hardware or software may obtain the contents of the translation table is specified in
 1306 [Translation of 64-bit Tag Encoding Company Prefix Indices Into EAN.UCC Company
 1307 Prefixes].

1308 *Individual Asset Reference* is a unique number for each instance. The 64-bit and 96-bit
 1309 tag encodings are only capable of representing a subset of asset references allowed in the
 1310 General EAN.UCC Specifications. The capacity of this asset reference is less than the
 1311 maximum EAN.UCC System specification for asset references, no leading zeros are
 1312 permitted, and only numbers are permitted.

1313 **3.8.1.1 GIAI-64 Encoding Procedure**

1314 The following procedure creates a GIAI-64 encoding.

1315 Given:

1316 An EAN.UCC GIAI consisting of digits $d_1d_2\dots d_K$ where $K \leq 30$.

1317 The length L of the company prefix portion of the GIAI

1318 A Filter Value F where $0 \leq F < 8$

1319 Procedure:

1320 1. Extract the EAN.UCC Company Prefix $d_1d_2\dots d_L$

1321 2. Do a reverse lookup of the Company Prefix in the Company Prefix Translation Table
1322 to obtain the corresponding Company Prefix Index, C . If the Company Prefix was not
1323 found in the Company Prefix Translation Table, stop: this GIAI cannot be encoded in the
1324 GIAI-64 encoding.

1325 3. Construct the Individual Asset Reference by concatenating digits $d_{(L+1)}d_{(L+2)}\dots d_K$. If
1326 any of these characters is not a digit, stop: this GIAI cannot be encoded in the GIAI-64
1327 encoding. Otherwise, consider the result to be a decimal integer, I . If $I \geq 2^{39}$, stop: this
1328 GIAI cannot be encoded in the GIAI-64 encoding. Also, if $K > L+1$ and $d_{(L+1)} = 0$, stop:
1329 this GIAI cannot be encoded in the GIAI-64 encoding (because leading zeros are not
1330 permitted except in the case where the Individual Asset Reference consists of a single
1331 zero digit).

1332 4. Construct the final encoding by concatenating the following bit fields, from most
1333 significant to least significant: Header 00001011 (8 bits), Filter Value F (3 bits),
1334 Company Prefix Index C from Step 2 (14 bits), Individual Asset Reference from Step 3
1335 (39 bits).

1336 **3.8.1.2 GIAI-64 Decoding Procedure**

1337 Given:

1338 An GIAI-64 as a 64-bit bit string 00001011 $b_{55}b_{54}\dots b_0$ (where the first eight bits
1339 00001011 are the header)

1340 Yields:

1341 An EAN.UCC GIAI

1342 A Filter Value

1343 Procedure:

1344 1. Bits $b_{55}b_{54}b_{53}$, considered as an unsigned integer, are the Filter Value.

1345 2. Extract the Company Prefix Index C by considering bits $b_{52}b_{51}\dots b_{39}$ as an unsigned
1346 integer.

1347 3. Look up the Company Prefix Index C in the Company Prefix Translation Table to
1348 obtain the EAN.UCC Company Prefix $p_1p_2\dots p_L$ consisting of L decimal digits (the value

1349 of L is also obtained from the table). If the Company Prefix Index C is not found in the
 1350 Company Prefix Translation Table, stop: this bit string cannot be decoded as a GIAI-64.

1351 4. Consider bits $b_{38}b_{37}...b_0$ as an unsigned integer. If this integer is greater than or equal
 1352 to $10^{(30-L)}$, stop: the input bit string is not a legal GIAI-64 encoding. Otherwise, convert
 1353 this integer to a decimal number $s_1s_2...s_J$, with no leading zeros (exception: if the integer
 1354 is equal to zero, convert it to a single zero digit).

1355 5. Construct a K-digit number $d_1d_2...d_K$ where $d_1d_2...d_L = p_1p_2...p_L$ from Step 3, and
 1356 $d_{(L+1)}d_{(L+2)}...d_K = s_1s_2...s_J$ from Step 4. This K-digit number, where $K \leq 30$, is the
 1357 EAN.UCC GIAI.

1358 **3.8.2 GIAI-96**

1359 In addition to a Header, the EPC GIAI-96 is composed of four fields: the *Filter Value*,
 1360 *Partition*, *Company Prefix*, and *Individual Asset Reference*, as shown in Table 22.

1361

	Header	Filter Value	Partition	Company Prefix	Individual Asset Reference
GIAI-96	8	3	3	20-40	62-42
	0011 0100 (Binary value)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,9 99,999 (Decimal capacity*)	4,611,686,018,427, 387,904 – 4,398,046,511,103 (Decimal capacity*)

1362

1363 *Capacity of Company Prefix and Individual Asset Reference fields vary according to contents of the
 1364 Partition field.

1365 **Table 22.** The EPC 96-bit GIAI bit allocation, header, and decimal capacity.

1366 *Header* is 8-bits, with a binary value of 0011 0100.

1367 *Filter Value* is not part of the GIAI or EPC identifier, but is used for fast filtering and
 1368 pre-selection of basic asset types. The Filter Values for 64-bit and 96-bit GIAI are the
 1369 same. See Table 21.

1370 The *Partition* is an indication of where the subsequent Company Prefix and Individual
 1371 Asset Reference numbers are divided. This organization matches the structure in the
 1372 EAN.UCC GIAI in which the Company Prefix may vary from 6 to 12 digits. The
 1373 available values of *Partition* and the corresponding sizes of the *Company Prefix* and
 1374 *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

Table 23. GIAI-96 Partitions.

1375

1376 *Company Prefix* contains a literal embedding of the Company Prefix.

1377 *Individual Asset Reference* is a unique number for each instance. The EPC representation
 1378 is only capable of representing a subset of asset references allowed in the General
 1379 EAN.UCC Specifications. The capacity of this asset reference is less than the maximum
 1380 EAN.UCC System specification for asset references, no leading zeros are permitted, and
 1381 only numbers are permitted.

1382 **3.8.2.1 GIAI-96 Encoding Procedure**

1383 The following procedure creates a GIAI-96 encoding.

1384 Given:

1385 An EAN.UCC GIAI consisting of digits $d_1d_2\dots d_K$, where $K \leq 30$.

1386 The length L of the Company Prefix portion of the GIAI

1387 A Filter Value F where $0 \leq F < 8$

1388 Procedure:

1389 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column
 1390 of the Partition Table (Table 23) to determine the Partition Value, P , the number of bits
 1391 M in the Company Prefix field, and the number of bits N in the Individual Asset
 1392 Reference field. If L is not found in any row of Table 23, stop: this GIAI cannot be
 1393 encoded in a GIAI-96.

1394 2. Construct the Company Prefix by concatenating digits $d_1d_2\dots d_L$ and considering the
 1395 result to be a decimal integer, C .

1396 3. Construct the Individual Asset Reference by concatenating digits $d_{(L+1)}d_{(L+2)}\dots d_K$. If
 1397 any of these characters is not a digit, stop: this GIAI cannot be encoded in the GIAI-96
 1398 encoding. Otherwise, consider the result to be a decimal integer, S . If $S \geq 2^N$, stop: this

1399 GIAI cannot be encoded in the GIAI-96 encoding. Also, if $K > L+1$ and $d_{(L+1)} = 0$, stop:
1400 this GIAI cannot be encoded in the GIAI-96 encoding (because leading zeros are not
1401 permitted except in the case where the Individual Asset Reference consists of a single
1402 zero digit).

1403 4. Construct the final encoding by concatenating the following bit fields, from most
1404 significant to least significant: Header 00110100 (8 bits), Filter Value F (3 bits),
1405 Partition Value P from Step 2 (3 bits), Company Prefix C from Step 3 (M bits),
1406 Individual Asset Number S from Step 4 (N bits). Note that $M+N = 82$ bits for all P .

1407 3.8.2.2 GIAI-96 Decoding Procedure

1408 Given:

1409 A GIAI-96 as a 96-bit bit string 00110100 $b_{87}b_{86} \dots b_0$ (where the first eight bits
1410 00110100 are the header)

1411 Yields:

1412 An EAN.UCC GIAI

1413 A Filter Value

1414 Procedure:

- 1415 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.
- 1416 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
1417 $P = 7$, stop: this bit string cannot be decoded as a GIAI-96.
- 1418 3. Look up the Partition Value P in Table 23 to obtain the number of bits M in the
1419 Company Prefix and the number of digits L in the Company Prefix.
- 1420 4. Extract the Company Prefix C by considering bits $b_{81}b_{80} \dots b_{(82-M)}$ as an unsigned
1421 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a
1422 legal GIAI-96 encoding. Otherwise, convert this integer into a decimal number $p_1p_2 \dots p_L$,
1423 adding leading zeros as necessary to make up L digits in total.
- 1424 5. Extract the Individual Asset Reference by considering bits $b_{(81-M)}b_{(80-M)} \dots b_0$ as an
1425 unsigned integer. If this integer is greater than or equal to $10^{(30-L)}$, stop: the input bit
1426 string is not a legal GIAI-96 encoding. Otherwise, convert this integer to a decimal
1427 number $s_1s_2 \dots s_J$, with no leading zeros (exception: if the integer is equal to zero, convert
1428 it to a single zero digit).
- 1429 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
1430 $d_{(L+1)}d_{(L+2)} \dots d_K = s_1s_2 \dots s_J$ from Step 5. This K -digit number, where $K \leq 30$, is the
1431 EAN.UCC GIAI.

1432 4 URI Representation

1433 This section defines standards for the encoding of the Electronic Product Code™ as a
1434 Uniform Resource Identifier (URI). The URI Encoding complements the EPC Tag
1435 Encodings defined for use within RFID tags and other low-level architectural

1436 components. URIs provide a means for application software to manipulate Electronic
1437 Product Codes in a way that is independent of any particular tag-level representation,
1438 decoupling application logic from the way in which a particular Electronic Product Code
1439 was obtained from a tag.

1440 This section defines four categories of URI. The first are URIs for pure identities,
1441 sometimes called “canonical forms.” These contain only the unique information that
1442 identifies a specific physical object, and are independent of tag encodings. The second
1443 category are URIs that represent specific tag encodings. These are used in software
1444 applications where the encoding scheme is relevant, as when commanding software to
1445 write a tag. The third category are URIs that represent patterns, or sets of EPCs. These
1446 are used when instructing software how to filter tag data. The last category is a URI
1447 representation for raw tag information, generally used only for error reporting purposes.

1448 All categories of URIs are represented as Uniform Reference Names (URNs) as defined
1449 by [RFC2141], where the URN Namespace is `epc`.

1450 This section complements Section 3, EPC Bit-level Encodings, which specifies the
1451 currently defined tag-level representations of the Electronic Product Code.

1452 **4.1 URI Forms for Pure Identities**

1453 (This section is non-normative; the formal specifications for the URI types are given in
1454 Sections 4.3 and 5.)

1455 URI forms are provided for pure identities, which contain just the EPC fields that serve to
1456 distinguish one object from another. These URIs take the form of Universal Resource
1457 Names (URNs), with a different URN namespace allocated for each pure identity type.

1458 For the EPC General Identifier (Section 2.1.1), the pure identity URI representation is as
1459 follows:

1460 `urn:epc:id:gid:GeneralManagerNumber.ObjectClass.SerialNumber`

1461 In this representation, the three fields *GeneralManagerNumber*, *ObjectClass*,
1462 and *SerialNumber* correspond to the three components of an EPC General Identifier
1463 as described in Section 2.1.1. In the URI representation, each field is expressed as a
1464 decimal integer, with no leading zeros (except where a field’s value is equal to zero, in
1465 which case a single zero digit is used).

1466 There are also pure identity URI forms defined for identity types corresponding to certain
1467 types within the EAN.UCC System family of codes as defined in Section 2.1.2; namely,
1468 the Serialized Global Trade Item Number (SGTIN), the Serial Shipping Container Code
1469 (SSCC), the Serialized Global Location Number (SGLN), the Global Reusable Asset
1470 Identifier (GRAI), and the Global Individual Asset Identifier (GIAI). The URI
1471 representations corresponding to these identifiers are as follows:

1472 `urn:epc:id:sgtin:CompanyPrefix.ItemReference.SerialNumber`

1473 `urn:epc:id:sscc:CompanyPrefix.SerialReference`

1474 `urn:epc:id:sgln:CompanyPrefix.LocationReference.SerialNumber`

1475 urn:epc:id:grai:CompanyPrefix.AssetType.SerialNumber
1476 urn:epc:id:giai:CompanyPrefix.IndividualAssetReference
1477 In these representations, *CompanyPrefix* corresponds to an EAN.UCC company
1478 prefix assigned to a manufacturer by the UCC or EAN. (A UCC company prefix is
1479 converted to an EAN.UCC company prefix by adding one leading zero at the beginning.)
1480 The number of digits in this field is significant, and leading zeros are included as
1481 necessary.

1482 The *ItemReference*, *SerialReference*, *LocationReference*, and
1483 *AssetType* fields correspond to the similar fields of the GTIN, SSCC, GLN, and GRAI,
1484 respectively. Like the *CompanyPrefix* field, the number of digits in these fields is
1485 significant, and leading zeros are included as necessary. The number of digits in these
1486 fields, when added to the number of digits in the *CompanyPrefix* field, always total
1487 the same number of digits according to the identity type: 13 digits total for SGTIN, 17
1488 digits total for SSCC, 12 digits total for SGLN, and 12 characters total for the GRAI.
1489 (The *ItemReference* field of the SGTIN includes the GTIN Indicator (PI) digit,
1490 appended to the beginning of the item reference. The *SerialReference* field
1491 includes the SSCC Extension Digit (ED), appended at the beginning of the serial
1492 reference. In no case are check digits included in URI representations.)

1493 In contrast to the other fields, the *SerialNumber* field of the SGLN is a pure integer,
1494 with no leading zeros. The *SerialNumber* field of the SGTIN and GRAI, as well as
1495 the *IndividualAssetReference* field of the GIAI, may include digits, letters, and
1496 certain other characters. In order for an SGTIN, GRAI, or GIAI to be encodable on a 64-
1497 bit and 96-bit tag, however, these fields must consist only of digits with no leading zeros.
1498 These restrictions are defined in the encoding procedures for these types, as well as in
1499 Appendix F.

1500 An SGTIN, SSCC, etc in this form is said to be in SGTIN-URI form, SSCC-URI form,
1501 etc form, respectively. Here are examples:

1502 urn:epc:id:sgtin:0652642.800031.400
1503 urn:epc:id:sscc:0652642.0123456789
1504 urn:epc:id:sgln:0652642.12345.400
1505 urn:epc:id:grai:0652642.12345.1234
1506 urn:epc:id:giai:0652642.123456

1507 Referring to the first example, the corresponding GTIN-14 code is 80652642000311.
1508 This divides as follows: the first digit (8) is the PI digit, which appears as the first digit
1509 of the *ItemReference* field in the URI, the next seven digits (0652642) are the
1510 *CompanyPrefix*, the next five digits (00031) are the remainder of the
1511 *ItemReference*, and the last digit (1) is the check digit, which is not included in the
1512 URI.

1513 Referring to the second example, the corresponding SSCC is 006526421234567896 and
1514 the last digit (6) is the check digit, not included in the URI.

1515 Referring to the third example, the corresponding GLN is 0652642123458, where the last
1516 digit (8) is the check digit, not included in the URI.

1517 Referring to the fourth example, the corresponding GRAI is 006526421234581234,
1518 where the digit (8) is the check digit, not included in the URI.

1519 Referring to the fifth example, the corresponding GIAI is 0652642123456. (GIAI codes
1520 do not include a check digit.)

1521 Note that all five URI forms have an explicit indication of the division between the
1522 company prefix and the remainder of the code. This is necessary so that the URI
1523 representation may be converted into tag encodings. In general, the URI representation
1524 may be converted to the corresponding EAN.UCC numeric form (by combining digits
1525 and calculating the check digit), but converting from the EAN.UCC numeric form to the
1526 corresponding URI representation requires independent knowledge of the length of the
1527 company prefix.

1528 **4.2 URI Forms for Related Data Types**

1529 (This section is non-normative; the formal specifications for the URI types are given in
1530 Sections 4.3 and 5.)

1531 There are several data types that commonly occur in applications that manipulate
1532 Electronic Product Codes, which are not themselves Electronic Product Codes but are
1533 closely related. This specification provides URI forms for those as well. The general
1534 form of the *epc* URN Namespace is

1535 `urn:epc:type:typeSpecificPart`

1536 The *type* field identifies a particular data type, and *typeSpecificPart* encodes
1537 information appropriate for that data type. Currently, there are three possibilities defined
1538 for *type*, discussed in the next three sections.

1539 **4.2.1 URIs for EPC Tags**

1540 In some cases, it is desirable to encode in URI form a specific tag encoding of an EPC.
1541 For example, an application may wish to report to an operator what kinds of tags have
1542 been read. In another example, an application responsible for programming tags needs to
1543 be told not only what Electronic Product Code to put on a tag, but also the encoding
1544 scheme to be used. Finally, applications that wish to manipulate any additional data
1545 fields on tags need some representation other than the pure identity forms.

1546 EPC Tag URIs are encoded by setting the *type* field to *tag*, with the entire URI having
1547 this form:

1548 `urn:epc:tag:EncName:EncodingSpecificFields`

1549 where *EncName* is the name of an EPC encoding scheme, and
1550 *EncodingSpecificFields* denotes the data fields required by that encoding
1551 scheme, separated by dot characters. Exactly what fields are present depends on the
1552 specific encoding scheme used.

1591 **4.2.3 URIs for EPC Patterns**

1592 Certain software applications need to specify rules for filtering lists of EPCs according to
1593 various criteria. This specification provides a *pattern* URI form for this purpose. A
1594 pattern URI does not represent a single Electronic Product Code, but rather refers to a set
1595 of EPCs. A typical pattern looks like this:

1596 `urn:epc:pat:sgtin-64:3.0652642.[1024-2047].*`

1597 This pattern refers to any EPC SGTIN Identifier 64-bit tag, whose Filter field is 3, whose
1598 Company Prefix is 0652642, whose Item Reference is in the range $1024 \leq \textit{itemReference}$
1599 ≤ 2047 , and whose Serial Number may be anything at all.

1600 In general, there is a pattern form corresponding to each tag encoding form
1601 (Section 4.2.1), whose syntax is essentially identical except that ranges or the star (*)
1602 character may be used in each field.

1603 For the SGTIN, SSCC, and SGLN patterns, the pattern syntax slightly restricts how
1604 wildcards and ranges may be combined. Only two possibilities are permitted for the
1605 *CompanyPrefix* field. One, it may be a star (*), in which case the following field
1606 (*ItemReference*, *SerialReference*, or *LocationReference*) must also be a
1607 star. Two, it may be a specific company prefix, in which case the following field may be
1608 a number, a range, or a star. A range may not be specified for the *CompanyPrefix*.

1609 *Explanation (non-normative): Because the company prefix is variable length, a range*
1610 *may not be specified, as the range might span different lengths. Also, in the case of the*
1611 *SGTIN-64, SSCC-64, and GLN-64 encodings, the tag contains a manager index which*
1612 *maps into a company prefix but not in a way that preserves contiguous ranges. When a*
1613 *particular company prefix is specified, however, it is possible to match ranges or all*
1614 *values of the following field, because its length is fixed for a given company prefix. The*
1615 *other case that is allowed is when both fields are a star, which works for all tag*
1616 *encodings because the corresponding tag fields (including the Partition field, where*
1617 *present) are simply ignored.*

1618 **4.3 Syntax**

1619 The syntax of the EPC-URI and the URI forms for related data types are defined by the
1620 following grammar.

1621 **4.3.1 Common Grammar Elements**

1622 `NumericComponent ::= ZeroComponent | NonZeroComponent`

1623 `ZeroComponent ::= "0"`

1624 `NonZeroComponent ::= NonZeroDigit Digit*`

1625 `PaddedNumericComponent ::= Digit+`

1626 `Digit ::= "0" | NonZeroDigit`

1627 `NonZeroDigit ::= "1" | "2" | "3" | "4"`
1628 `| "5" | "6" | "7" | "8" | "9"`

```

1629 UpperAlpha ::= "A" | "B" | "C" | "D" | "E" | "F" | "G"
1630             | "H" | "I" | "J" | "K" | "L" | "M" | "N"
1631             | "O" | "P" | "Q" | "R" | "S" | "T" | "U"
1632             | "V" | "W" | "X" | "Y" | "Z"
1633 LowerAlpha ::= "a" | "b" | "c" | "d" | "e" | "f" | "g"
1634             | "h" | "i" | "j" | "k" | "l" | "m" | "n"
1635             | "o" | "p" | "q" | "r" | "s" | "t" | "u"
1636             | "v" | "w" | "x" | "y" | "z"
1637 OtherChar ::= "!" | "'" | "(" | ")" | "*" | "+" | "," | "-"
1638             | "." | ":" | ";" | "=" | "_"
1639 Escape ::= "%" HexChar HexChar
1640 HexChar ::= Digit | "A" | "B" | "C" | "D" | "E" | "F"
1641             | "a" | "b" | "c" | "d" | "e" | "f"
1642 GS3A3Char ::= Digit | UpperAlpha | LowerAlpha | OtherChar
1643             | Escape
1644 GS3A3Component ::= GS3A3Char+

```

1645 The syntactic construct `GS3A3Component` is used to represent fields of EAN.UCC
1646 codes that permit alphanumeric and other characters as specified in Figure 3A3-1 of the
1647 EAN.UCC General Specifications. Owing to restrictions on URN syntax as defined by
1648 [RFC2141], not all characters permitted in the EAN.UCC General Specifications may be
1649 represented directly in a URN. Specifically, the characters " (double quote), % (percent),
1650 & (ampersand), / (forward slash), < (less than), > (greater than), and ? (question mark)
1651 are permitted in the General Specifications but may not be included directly in a URN.
1652 To represent one of these characters in a URN, escape notation must be used in which the
1653 character is represented by a percent sign, followed by two hexadecimal digits that give
1654 the ASCII character code for the character.

1655 4.3.2 EPCGID-URI

```

1656 EPCGID-URI ::= "urn:epc:id:gid:" 2*(NumericComponent ".")
1657 NumericComponent

```

1658 4.3.3 SGTIN-URI

```

1659 SGTIN-URI ::= "urn:epc:id:sgtin:" SGTINURIBody
1660 SGTINURIBody ::= 2*(PaddedNumericComponent ".")
1661 GS3A3Component

```

1662 The number of characters in the two `PaddedNumericComponent` fields must total 13
1663 (not including any of the dot characters).

1664 The Serial Number field of the SGTIN-URI is expressed as a `GS3A3Component`,
1665 which permits the representation of all characters permitted in the UCC/EAN-128
1666 Application Identifier 21 Serial Number according to the EAN.UCC General

1667 Specifications. SGTIN-URIs that are derived from 64-bit and 96-bit tag encodings,
1668 however, will have Serial Numbers that consist only of digit characters and which have
1669 no leading zeros. These limitations are described in the encoding procedures, and in
1670 Appendix F.

1671 **4.3.4 SSCC-URI**

1672 SSCC-URI ::= "urn:epc:id:sscc:" SSCCURIBody

1673 SSCCURIBody ::= PaddedNumericComponent "."

1674 PaddedNumericComponent

1675 The number of characters in the two PaddedNumericComponent fields must total 17
1676 (not including any of the dot characters).

1677 **4.3.5 SGLN-URI**

1678 SGLN-URI ::= "urn:epc:id:sgln:" SGLNURIBody

1679 SGLNURIBody ::= 2*(PaddedNumericComponent ".")

1680 NumericComponent

1681 The number of characters in the two PaddedNumericComponent fields must total 12
1682 (not including any of the dot characters).

1683 **4.3.6 GRAI-URI**

1684 GRAI-URI ::= "urn:epc:id:grai:" GRAIURIBody

1685 GRAIURIBody ::= 2*(PaddedNumericComponent ".")

1686 GS3A3Component

1687 The number of characters in the two PaddedNumericComponent fields must total 12
1688 (not including any of the dot characters).

1689 The Serial Number field of the GRAI-URI is expressed as a GS3A3Component, which
1690 permits the representation of all characters permitted in the Serial Number field of the
1691 GRAI according to the EAN.UCC General Specifications. GRAI-URIs that are derived
1692 from 64-bit and 96-bit tag encodings, however, will have Serial Numbers that consist
1693 only of digit characters and which have no leading zeros. These limitations are described
1694 in the encoding procedures, and in Appendix F.

1695 **4.3.7 GIAI-URI**

1696 GIAI-URI ::= "urn:epc:id:giai:" GIAIURIBody

1697 GIAIURIBody ::= PaddedNumericComponent "." GS3A3Component

1698 The total number of characters in the PaddedNumericComponent and
1699 GS3A3Component fields must not exceed 30 (not including the dot character that
1700 separates the two fields).

1701 The Individual Asset Reference field of the GIAI-URI is expressed as a
1702 GS3A3Component, which permits the representation of all characters permitted in the
1703 Individual Asset Reference field of the GIAI according to the EAN.UCC General
1704 Specifications. GIAI-URIs that are derived from 64-bit and 96-bit tag encodings,
1705 however, will have Individual Asset References that consist only of digit characters and
1706 which have no leading zeros. These limitations are described in the encoding procedures,
1707 and in Appendix F.

1708 **4.3.8 EPC Tag URI**

1709 TagURI ::= "urn:epc:tag:" TagURIBody
1710 TagURIBody ::= GIDTagURIBody | SGTINSGLNGRAITagURIBody |
1711 SSCCGIAITagURIBody
1712 GIDTagURIBody ::= GIDTagEncName ":" 2*(NumericComponent ".")
1713 NumericComponent
1714 GIDTagEncName ::= "gid-96"
1715 SGTINSGLNGRAITagURIBody ::= SGTINSGLNGRAITagEncName ":"
1716 NumericComponent ".") 2*(PaddedNumericComponent ".")
1717 NumericComponent
1718 SGTINSGLNGRAITagEncName ::= "sgtin-96" | "sgtin-64" | "sgln-
1719 96" | "sgln-64" | "grai-96" | "grai-64"
1720 SSCCGIAITagURIBody ::= SSCCGIAITagEncName ":"
1721 NumericComponent 2*("." PaddedNumericComponent)
1722 SSCCGIAITagEncName ::= "sscc-96" | "sscc-64" | "giai-96" |
1723 "giai-64"

1724 **4.3.9 Raw Tag URI**

1725 RawURI ::= "urn:epc:raw:" RawURIBody
1726 RawURIBody ::= NonZeroComponent "." NumericComponent

1727 **4.3.10 EPC Pattern URI**

1728 PatURI ::= "urn:epc:pat:" PatBody
1729 PatBody ::= GIDPatURIBody | SGTINSGLNGRAIPatURIBody |
1730 SSCCGIAIPatURIBody
1731 GIDPatURIBody ::= GIDTagEncName ":" 2*(PatComponent ".")
1732 PatComponent
1733 SGTINSGLNGRAIPatURIBody ::= SGTINSGLNGRAITagEncName ":"
1734 PatComponent "." GS1PatBody "." PatComponent
1735 SSCCGIAIPatURIBody ::= SSCCGIAITagEncName ":" PatComponent
1736 "." GS1PatBody

```

1737 GS1PatBody ::= "*" | ( PaddedNumericComponent "."
1738 PatComponent )
1739 PatComponent ::= NumericComponent
1740                 | StarComponent
1741                 | RangeComponent
1742 StarComponent ::= "*"
1743 RangeComponent ::= "[" NumericComponent "-"
1744                   NumericComponent "]"
1745 For a RangeComponent to be legal, the numeric value of the first
1746 NumericComponent must be less than or equal to the numeric value of the second
1747 NumericComponent.

```

1748 **4.3.11 Summary (non-normative)**

1749 The syntax rules above can be summarized informally as follows:

```

1750 urn:epc:id:gid:MMM.CCC.SSS
1751 urn:epc:id:sgtin:PPP.III.SSS
1752 urn:epc:id:sscc:PPP.III
1753 urn:epc:id:sgln:PPP.III
1754 urn:epc:id:grai:PPP.III.SSS
1755 urn:epc:id:giai:PPP.SSS
1756
1757 urn:epc:tag:sgtin-64:FFF.PPP.III.SSS
1758 urn:epc:tag:sscc-64:FFF.PPP.III
1759 urn:epc:tag:sgln-64:FFF.PPP.III.SSS
1760 urn:epc:tag:grai-64:FFF.PPP.III.SSS
1761 urn:epc:tag:giai-64:FFF.PPP.SSS
1762 urn:epc:tag:gid-96:MMM.CCC.SSS
1763 urn:epc:tag:sgtin-96:FFF.PPP.III.SSS
1764 urn:epc:tag:sscc-96:FFF.PPP.III
1765 urn:epc:tag:sgln-96:FFF.PPP.III.SSS
1766 urn:epc:tag:grai-96:FFF.PPP.III.SSS
1767 urn:epc:tag:giai-96:FFF.PPP.SSS
1768
1769 urn:epc:raw:LLL.BBB
1770

```

1771 urn:epc:pat:sgtin-64:*FFFpat.PPP.IIIpat.SSSpat*

1772 urn:epc:pat:sgtin-64:*FFFpat.*.*.SSSpat*

1773 urn:epc:pat:sscc-64:*FFFpat.PPP.IIIpat*

1774 urn:epc:pat:sscc-64:*FFFpat.*.**

1775 urn:epc:pat:sgln-64:*FFFpat.PPP.IIIpat.SSSpat*

1776 urn:epc:pat:sgln-64:*FFFpat.*.*.SSSpat*

1777 urn:epc:pat:grai-64:*FFFpat.PPP.IIIpat.SSSpat*

1778 urn:epc:pat:grai-64:*FFFpat.*.*.SSSpat*

1779 urn:epc:pat:giai-64:*FFFpat.PPP.SSSpat*

1780 urn:epc:pat:giai-64:*FFFpat.*.**

1781 urn:epc:pat:gid-96:*MMMpat.CCCpat.SSSpat*

1782 urn:epc:pat:sgtin-96:*FFFpat.PPP.IIIpat.SSSpat*

1783 urn:epc:pat:sgtin-96:*FFFpat.*.*.SSSpat*

1784 urn:epc:pat:sscc-96:*FFFpat.PPP.IIIpat*

1785 urn:epc:pat:sscc-96:*FFFpat.*.**

1786 urn:epc:pat:sgln-96:*FFFpat.PPP.IIIpat.SSSpat*

1787 urn:epc:pat:sgln-96:*FFFpat.*.*.SSSpat*

1788 urn:epc:pat:grai-96:*FFFpat.PPP.IIIpat.SSSpat*

1789 urn:epc:pat:grai-96:*FFFpat.*.*.SSSpat*

1790 urn:epc:pat:giai-96:*FFFpat.PPP.SSSpat*

1791 urn:epc:pat:giai-96:*FFFpat.*.**

1792 where

1793 *MMM* denotes a General Manager Number

1794 *CCC* denotes an Object Class number

1795 *SSS* denotes a Serial Number or GIAI Individual Asset Reference

1796 *PPP* denotes an EAN.UCC Company Prefix

1797 *III* denotes an SGTIN Item Reference (with Indicator Digit appended to the

1798 beginning), an SSCC Shipping Container Serial Number (with the Extension (ED) digit

1799 appended at the beginning), a SGLN Location Reference, or a GRAI Asset Type.

1800 *FFF* denotes a filter code as used by the SGTIN, SSCC, SGLN, GRAI, and GIAI tag

1801 encodings

1802 *XXXpat* is the same as *XXX* but allowing * and [lo-hi] pattern syntax in addition

1803 *LLL* denotes the number of bits of an uninterpreted bit sequence

1804 *BBB* denotes the literal value of an uninterpreted bit sequence converted to decimal
1805 and where all numeric fields are in decimal with no leading zeros (unless the overall
1806 value of the field is zero, in which case it is represented with a single 0 character).
1807 Exception: the length of *PPP* and *III* is significant, and leading zeros are used as
1808 necessary. The length of *PPP* is the length of the company prefix as assigned by EAN or
1809 UCC. The length of *III* plus the length of *PPP* must equal 13 for SGTIN, 17 for SSCC,
1810 12 for GLN, or 12 for GRAI.

1811 **5 Translation between EPC-URI and Other EPC** 1812 **Representations**

1813 This section defines the semantics of EPC-URI encodings, by defining how they are
1814 translated into other EPC encodings and vice versa.

1815 The following procedure translates a bit-level encoding of an EPC into an EPC-URI:

- 1816 1. Determine the identity type and encoding scheme by finding the row in Table 1
1817 (Section 3.1) that matches the most significant bits of the bit string. If the most
1818 significant bits do not match any row of the table, stop: the bit string is invalid
1819 and cannot be translated into an EPC-URI. Otherwise, if the encoding scheme is
1820 SGTIN-64 or SGTIN-96, proceed to Step 2; if the encoding scheme is SSCC-64
1821 or SSCC-96, proceed to Step 5; if the encoding scheme is SGLN-64 or SGLN-96,
1822 proceed to Step 8; if the encoding scheme is GRAI-64 or GRAI-96, proceed to
1823 Step 11; if the encoding scheme is GIAI-64 or GIAI-96, proceed to Step 14; if the
1824 encoding scheme is GID-96, proceed to Step 17.
- 1825 2. Follow the decoding procedure given in Section 3.4.1.2 (for SGTIN-64) or in
1826 Section 3.4.2.2 (for SGTIN-96) to obtain the decimal Company Prefix $p_1p_2\dots p_L$,
1827 the decimal Item Reference and Indicator $i_1i_2\dots i_{(13-L)}$, and the Serial Number S . If
1828 the decoding procedure fails, stop: the bit-level encoding cannot be translated into
1829 an EPC-URI.
- 1830 3. Create an EPC-URI by concatenating the following: the string
1831 `urn:epc:id:sgtin:`, the Company Prefix $p_1p_2\dots p_L$ where each digit
1832 (including any leading zeros) becomes the corresponding ASCII digit character, a
1833 dot (.) character, the Item Reference and Indicator $i_1i_2\dots i_{(13-L)}$ (handled similarly),
1834 a dot (.) character, and the Serial Number S as a decimal integer. The portion
1835 corresponding to the Serial Number must have no leading zeros, except where the
1836 Serial Number is itself zero in which case the corresponding URI portion must
1837 consist of a single zero character.
- 1838 4. Go to Step 19.
- 1839 5. Follow the decoding procedure given in Section 3.5.1.2 (for SSCC-64) or in
1840 Section 3.5.2.2 (for SSCC-96) to obtain the decimal Company Prefix $p_1p_2\dots p_L$,
1841 and the decimal Serial Reference $s_1s_2\dots s_{(17-L)}$. If the decoding procedure fails,
1842 stop: the bit-level encoding cannot be translated into an EPC-URI.

- 1843 6. Create an EPC-URI by concatenating the following: the string
1844 `urn:epc:id:sscc:`, the Company Prefix $p_1p_2\dots p_L$ where each digit (including
1845 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
1846 character, and the Serial Reference $s_1s_2\dots s_{(17-L)}$ (handled similarly).
- 1847 7. Go to Step 19.
- 1848 8. Follow the decoding procedure given in Section 3.6.1.2 (for SGLN-64) or in
1849 Section 3.6.2.2 (for SGLN-96) to obtain the decimal Company Prefix $p_1p_2\dots p_L$,
1850 the decimal Location Reference $i_1i_2\dots i_{(12-L)}$, and the Serial Number S . If the
1851 decoding procedure fails, stop: the bit-level encoding cannot be translated into an
1852 EPC-URI.
- 1853 9. Create an EPC-URI by concatenating the following: the string
1854 `urn:epc:id:sgln:`, the Company Prefix $p_1p_2\dots p_L$ where each digit (including
1855 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
1856 character, the Location Reference $i_1i_2\dots i_{(12-L)}$ (handled similarly), a dot (.)
1857 character, and the Serial Number S as a decimal integer. The portion
1858 corresponding to the Serial Number must have no leading zeros, except where the
1859 Serial Number is itself zero in which case the corresponding URI portion must
1860 consist of a single zero character.
- 1861 10. Go to Step 19.
- 1862 11. Follow the decoding procedure given in Section 3.7.1.2 (for GRAI-64) or in
1863 Section 3.7.2.2 (for GRAI-96) to obtain the decimal Company Prefix $p_1p_2\dots p_L$, the
1864 decimal Asset Type $i_1i_2\dots i_{(12-L)}$, and the Serial Number S . If the decoding
1865 procedure fails, stop: the bit-level encoding cannot be translated into an EPC-URI.
- 1866 12. Create an EPC-URI by concatenating the following: the string
1867 `urn:epc:id:grai:`, the Company Prefix $p_1p_2\dots p_L$ where each digit (including
1868 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
1869 character, the Asset Type $i_1i_2\dots i_{(12-L)}$ (handled similarly), a dot (.) character, and
1870 the Serial Number S as a decimal integer. The portion corresponding to the Serial
1871 Number must have no leading zeros, except where the Serial Number is itself zero
1872 in which case the corresponding URI portion must consist of a single zero
1873 character.
- 1874 13. Go to Step 19.
- 1875 14. Follow the decoding procedure given in Section 3.8.1.2 (for GIAI-64) or in
1876 Section 3.8.2.2 (for GIAI-96) to obtain the decimal Company Prefix $p_1p_2\dots p_L$, and
1877 the Individual Asset Reference S . If the decoding procedure fails, stop: the bit-
1878 level encoding cannot be translated into an EPC-URI.
- 1879 15. Create an EPC-URI by concatenating the following: the string
1880 `urn:epc:id:giai:`, the Company Prefix $p_1p_2\dots p_L$ where each digit (including
1881 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
1882 character, and the Individual Asset Reference S as a decimal integer. The portion
1883 corresponding to the Individual Asset Reference must have no leading zeros,

1884 except where the Individual Asset Reference is itself zero in which case the
1885 corresponding URI portion must consist of a single zero character.

1886 16. Go to Step 19.

1887 17. Follow the decoding procedure given in Section 3.3.1.2 to obtain the General
1888 Manager Number M , the Object Class C , and the Serial Number S .

1889 18. Create an EPC-URI by concatenating the following: the string
1890 `urn:epc:id:gid:`, the General Manager Number as a decimal integer, a dot
1891 (`.`) character, the Object Class as a decimal integer, a dot (`.`) character, and the
1892 Serial Number S as a decimal integer. Each decimal number must have no
1893 leading zeros, except where the integer is itself zero in which case the
1894 corresponding URI portion must consist of a single zero character.

1895 19. The translation is now complete.

1896 The following procedure translates a bit-level tag encoding into either an EPC Tag URI
1897 or a Raw Tag URI:

1898 1. Determine the identity type and encoding scheme by finding the row in Table 1
1899 (Section 3.1) that matches the most significant bits of the bit string. If the
1900 encoding scheme is SGTIN-64 or SGTIN-96, proceed to Step 2; if the encoding
1901 scheme is SSCC-64 or SSCC-96, proceed to Step 5; if the encoding scheme is
1902 SGLN-64 or SGLN-96, proceed to Step 8; if the encoding scheme is GRAI-64 or
1903 GRAI-96, proceed to Step 11, if the encoding scheme is GIAI-64 or GIAI-96,
1904 proceed to Step 14, if the encoding scheme is GID-96, proceed to Step 17;
1905 otherwise, proceed to Step 20.

1906 2. Follow the decoding procedure given in Section 3.4.1.2 (for SGTIN-64) or in
1907 Section 3.4.2.2 (for SGTIN-96) to obtain the decimal Company Prefix $p_1p_2\dots p_L$,
1908 the decimal Item Reference and Indicator $i_1i_2\dots i_{(13-L)}$, the Filter Value F , and the
1909 Serial Number S . If the decoding procedure fails, proceed to Step 20, otherwise
1910 proceed to the next step.

1911 3. Create an EPC Tag URI by concatenating the following: the string
1912 `urn:epc:tag:`, the encoding scheme (`sgtin-64` or `sgtin-96`), a colon (`:`)
1913 character, the Filter Value F as a decimal integer, a dot (`.`) character, the
1914 Company Prefix $p_1p_2\dots p_L$ where each digit (including any leading zeros) becomes
1915 the corresponding ASCII digit character, a dot (`.`) character, the Item Reference
1916 and Indicator $i_1i_2\dots i_{(13-L)}$ (handled similarly), a dot (`.`) character, and the Serial
1917 Number S as a decimal integer. The portions corresponding to the Filter Value
1918 and Serial Number must have no leading zeros, except where the corresponding
1919 integer is itself zero in which case a single zero character is used.

1920 4. Go to Step 21.

1921 5. Follow the decoding procedure given in Section 3.5.1.2 (for SSCC-64) or in
1922 Section 3.5.2.2 (for SSCC-96) to obtain the decimal Company Prefix $p_1p_2\dots p_L$,
1923 and the decimal Serial Reference $i_1i_2\dots s_{(17-L)}$, and the Filter Value F . If the
1924 decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.

- 1925 6. Create an EPC Tag URI by concatenating the following: the string
1926 urn:epc:tag:, the encoding scheme (sscc-64 or sssc-96), a colon (:)
1927 character, the Filter Value F as a decimal integer, a dot (.) character, the
1928 Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes
1929 the corresponding ASCII digit character, a dot (.) character, and the Serial
1930 Reference $i_1i_2...i_{(17-L)}$ (handled similarly).
- 1931 7. Go to Step 21.
- 1932 8. Follow the decoding procedure given in Section 3.6.1.2 (for SGLN-64) or in
1933 Section 3.6.2.2 (for SGLN-96) to obtain the decimal Company Prefix $p_1p_2...p_L$,
1934 the decimal Location Reference $i_1i_2...i_{(12-L)}$, the Filter Value F , and the Serial
1935 Number S . If the decoding procedure fails, proceed to Step 20, otherwise proceed
1936 to the next step.
- 1937 9. Create an EPC Tag URI by concatenating the following: the string
1938 urn:epc:tag:, the encoding scheme (sgln-64 or sglN-96), a colon (:)
1939 character, the Filter Value F as a decimal integer, a dot (.) character, the
1940 Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes
1941 the corresponding ASCII digit character, a dot (.) character, the Location
1942 Reference $i_1i_2...i_{(12-L)}$ (handled similarly), a dot (.) character, and the Serial
1943 Number S as a decimal integer. The portions corresponding to the Filter Value
1944 and Serial Number must have no leading zeros, except where the corresponding
1945 integer is itself zero in which case a single zero character is used.
- 1946 10. Go to Step 21.
- 1947 11. Follow the decoding procedure given in Section 3.7.1.2 (for GRAI-64) or in
1948 Section 3.7.2.2 (for GRAI-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, the
1949 decimal Asset Type $i_1i_2...i_{(12-L)}$, the Filter Value F , and the Serial Number
1950 $d_1s_2...d_K$. If the decoding procedure fails, proceed to Step 20, otherwise proceed
1951 to the next step.
- 1952 12. Create an EPC Tag URI by concatenating the following: the string
1953 urn:epc:tag:, the encoding scheme (grai-64 or grai-96), a colon (:)
1954 character, the Filter Value F as a decimal integer, a dot (.) character, the
1955 Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes
1956 the corresponding ASCII digit character, a dot (.) character, the Asset Type
1957 $s_1s_2...s_{(12-L)}$ (handled similarly), a dot (.) character, and the Serial Number
1958 $d_1s_2...d_K$ as a decimal integer. The portions corresponding to the Filter Value
1959 and Serial Number must have no leading zeros, except where the corresponding
1960 integer is itself zero in which case a single zero character is used.
- 1961 13. Got to Step 21.
- 1962 14. Follow the decoding procedure given in Section 3.8.1.2 (for GIAI-64) or in
1963 Section 3.8.2.2 (for GIAI-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, the
1964 decimal Individual Asset Reference $s_1s_2...s_J$, and the Filter Value F . If the
1965 decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.

- 1966 15. Create an EPC Tag URI by concatenating the following: the string
 1967 urn:epc:tag:, the encoding scheme (giai-64 or giai-96), a colon (:)
 1968 character, the Filter Value *F* as a decimal integer, a dot (.) character, the Company
 1969 Prefix *p₁p₂...p_L* where each digit (including any leading zeros) becomes the
 1970 corresponding ASCII digit character, a dot (.) character, and the Individual Asset
 1971 Reference *i₁i₂...i_j* (handled similarly). The portion corresponding to the Filter
 1972 Value must have no leading zeros, except where the corresponding integer is itself
 1973 zero in which case a single zero character is used.
- 1974 16. Go to Step 21.
- 1975 17. Follow the decoding procedure given in Section 3.3.1.2 to obtain the EPC
 1976 Manager Number, the Object Class, and the Serial Number.
- 1977 18. Create an EPC Tag URI by concatenating the following: the string
 1978 urn:epc:tag:gid-96:, the General Manager Number as a decimal number,
 1979 a dot (.) character, the Object Class as a decimal number, a dot (.) character, and
 1980 the Serial Number as a decimal number. Each decimal number must have no
 1981 leading zeros, except where the integer is itself zero in which case the
 1982 corresponding URI portion must consist of a single zero character.
- 1983 19. Go to Step 21.
- 1984 20. This tag is not a recognized EPC encoding, therefore create an EPC Raw URI by
 1985 concatenating the following: the string urn:epc:raw:, the length of the bit
 1986 string, a dot (.) character, and the value of the bit string considered as a single
 1987 decimal integer. Both the length and the value must have no leading zeros, except
 1988 if the value is itself zero in which case a single zero character is used.
- 1989 21. The translation is now complete.

1990

1991 The following procedure translates a URI into a bit-level EPC:

- 1992 1. If the URI is an SGTIN-URI (urn:epc:id:sgtin:), an SSCC-URI
 1993 (urn:epc:id:sscc:), an SGLN-URI (urn:epc:id:sgln:), a GRAI-
 1994 URI (urn:epc:id:grai:), a GIAI-URI (urn:epc:id:giai:), a GID-
 1995 URI (urn:epc:id:gid:), or an EPC Pattern URI (urn:epc:pat:), the
 1996 URI cannot be translated into a bit-level EPC.
- 1997 2. If the URI is a Raw Tag URI (urn:epc:raw:), create the bit-level EPC by
 1998 converting the second component of the Raw Tag URI into a binary integer,
 1999 whose length is equal to the first component of the Raw Tag URI. If the value of
 2000 the second component is too large to fit into a binary integer of that size, the URI
 2001 cannot be translated into a bit-level EPC.
- 2002 3. If the URI is an EPC Tag URI (urn:epc:tag:encName:), parse the URI
 2003 using the grammar for TagURI as given in Section 4.3.8. If the URI cannot be
 2004 parsed using this grammar, stop: the URI is illegal and cannot be translated into a
 2005 bit-level EPC. Otherwise, if *encName* is *sgtin-96* or *sgtin-64* go to Step 4,

2006 if *encName* is *sscc-96* or *sscc-64* go to Step 9, if *encName* is *sgln-96*
2007 or *sgln-64* go to Step 13, if *encName* is *grai-96* or *grai-64* go to Step 18,
2008 if *encName* is *giai-96* or *giai-64* go to Step 22, or if *encName* is *gid-96*
2009 go to Step 26.

2010 4. Let the URI be written as
2011 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F . p_1 p_2 \dots p_L . i_1 i_2 \dots i_{(13-L)} . s_1 s_2 \dots s_S$.

2012 5. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .

2013 6. Interpret $s_1 s_2 \dots s_S$ as a decimal integer S .

2014 7. Carry out the encoding procedure defined in Section 3.4.1.1 (SGTIN-64) or
2015 Section 3.4.2.1 (SGTIN-96), using $i_1 p_1 p_2 \dots p_L i_2 \dots i_{(13-L)} 0$ as the EAN.UCC
2016 GTIN-14 (the trailing zero is a dummy check digit, which is ignored by the
2017 encoding procedure), L as the length of the EAN.UCC company prefix, F from
2018 Step 5 as the Filter Value, and S from Step 6 as the Serial Number. If the
2019 encoding procedure fails because an input is out of range, or because the
2020 procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.

2021 8. Go to Step 31.

2022 9. Let the URI be written as
2023 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F . p_1 p_2 \dots p_L . i_1 i_2 \dots i_{(17-L)}$.

2024 10. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .

2025 11. Carry out the encoding procedure defined in Section 3.5.1.1 (SSCC-64) or
2026 Section 3.5.2.1 (SSCC-96), using $i_1 p_1 p_2 \dots p_L i_2 i_3 \dots i_{(17-L)} 0$ as the EAN.UCC
2027 SSCC, L as the length of the EAN.UCC company prefix, and F from Step 10 as
2028 the Filter Value. If the encoding procedure fails because an input is out of range,
2029 or because the procedure indicates a failure, stop: this URI cannot be encoded
2030 into an EPC tag.

2031 12. Go to Step 31.

2032 13. Let the URI be written as
2033 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F . p_1 p_2 \dots p_L . i_1 i_2 \dots i_{(12-L)} . s_1 s_2 \dots s_S$.

2034 14. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .

2035 15. Interpret $s_1 s_2 \dots s_S$ as a decimal integer S .

2036 16. Carry out the encoding procedure defined in Section 3.6.1.1 (SGLN-64) or
2037 Section 3.6.2.1 (SGLN-96), using $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0$ as the EAN.UCC
2038 GLN (the trailing zero is a dummy check digit, which is ignored by the encoding
2039 procedure), L as the length of the EAN.UCC company prefix, F from Step 14 as
2040 the Filter Value, and S from Step 15 as the Serial Number. If the encoding
2041 procedure fails because an input is out of range, or because the procedure
2042 indicates a failure, stop: this URI cannot be encoded into an EPC tag.

2043 17. Go to Step 31.

- 2044 18. Let the URI be written as
 2045 $\text{urn:epc:tag:encName:f}_1\text{f}_2\dots\text{f}_F.p_1p_2\dots p_L.i_1i_2\dots i_{(12-L)}.s_1s_2\dots s_S.$
- 2046 19. Interpret $f_1f_2\dots f_F$ as a decimal integer F .
- 2047 20. Carry out the encoding procedure defined in Section 3.7.1.1 (GRAI-64) or
 2048 Section 3.7.2.1 (GRAI-96), using $0p_1p_2\dots p_Li_1i_2\dots i_{(12-L)}0s_1s_2\dots s_S$ as the
 2049 EAN.UCC GRAI (the second zero is a dummy check digit, which is ignored by
 2050 the encoding procedure), L as the length of the EAN.UCC company prefix, and F
 2051 from Step 19 as the Filter Value. If the encoding procedure fails because an input
 2052 is out of range, or because the procedure indicates a failure, stop: this URI cannot
 2053 be encoded into an EPC tag.
- 2054 21. Go to Step 31.
- 2055 22. Let the URI be written as
 2056 $\text{urn:epc:tag:encName:f}_1\text{f}_2\dots\text{f}_F.p_1p_2\dots p_L.s_1s_2\dots s_S.$
- 2057 23. Interpret $f_1f_2\dots f_F$ as a decimal integer F .
- 2058 24. Carry out the encoding procedure defined in Section 3.8.1.1 (GIAI-64) or
 2059 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
 2060 the length of the EAN.UCC company prefix, and F from Step 23 as the Filter
 2061 Value. If the encoding procedure fails because an input is out of range, or
 2062 because the procedure indicates a failure, stop: this URI cannot be encoded into
 2063 an EPC tag.
- 2064 25. Go to Step 31.
- 2065 26. Let the URI be written as
 2066 $\text{urn:epc:tag:encName:m}_1m_2\dots m_L.c_1c_2\dots c_K.s_1s_2\dots s_S.$
- 2067 27. Interpret $m_1m_2\dots m_L$ as a decimal integer M .
- 2068 28. Interpret $c_1c_2\dots c_K$ as a decimal integer C .
- 2069 29. Interpret $s_1s_2\dots s_S$ as a decimal integer S .
- 2070 30. Carry out the encoding procedure defined in Section 3.3.1.1 using M from Step 27
 2071 as the General Manager Number, C from Step 28 as the Object Class, and S from
 2072 Step 29 as the Serial Number. If the encoding procedure fails because an input is
 2073 out of range, or because the procedure indicates a failure, stop: this URI cannot
 2074 be encoded into an EPC tag.
- 2075 31. The translation is complete.

2076 6 Semantics of EPC Pattern URIs

2077 The meaning of an EPC Pattern URI (urn:epc:pat:) can be formally defined as
 2078 denoting a set of encoding-specific EPCs. The set of EPCs denoted by a specific EPC
 2079 Pattern URI is defined by the following decision procedure, which says whether a given
 2080 EPC Tag URI belongs to the set denoted by the EPC Pattern URI.

2081 Let $urn:epc:pat:EncName:P1.P2...Pn$ be an EPC Pattern URI. Let
 2082 $urn:epc:tag:EncName:C1.C2...Cn$ be an EPC Tag URI, where the *EncName*
 2083 field of both URIs is the same. The number of components (*n*) depends on the value of
 2084 *EncName*.

2085 First, any EPC Tag URI component *C_i* is said to *match* the corresponding EPC Pattern
 2086 URI component *P_i* if:

2087 *P_i* is a *NumericComponent*, and *C_i* is equal to *P_i*; or

2088 *P_i* is a *PaddedNumericComponent*, and *C_i* is equal to *P_i* both in numeric value as
 2089 well as in length; or

2090 *P_i* is a *RangeComponent* [*lo-hi*], and $lo \leq C_i \leq hi$; or

2091 *P_i* is a *StarComponent* (and *C_i* is anything at all)

2092 Then the EPC Tag URI is a member of the set denoted by the EPC Pattern URI if and
 2093 only if *C_i* matches *P_i* for all $1 \leq i \leq n$.

2094 **7 Background Information**

2095 This document represents the contributions of many people, especially the contributions
 2096 of the Tag Data Standards Group and the URI Representation Group.

2097 **EPC Tag Data Standards Group**

Bud Babcock	Procter & Gamble
Jason Carney	Ahold
Hal Charych	Symbol
Chris Cummins	Uniform Code Council
Falk Gernot	Metro
Yuichiro Hanawa	Mitsui, USA
Mark Harrison	Cambridge Auto-ID Lab
Larry Hilgert	Pepsico (Quaker)
André Frank	Sara Lee/DE
Jonathan Loretto	Cap Gemini Ernst & Young
Ron Moser	Wal-Mart
Don Mowery	Nestle
Steve Morris	Printronix
Bob Mytkowicz	Gillette
Doug Naal	Kraft
Juli Nackers	Kimberly-Clark
Robert Nonneman	UPS

Richard Probst	Nominum
Steve Rehling	Procter & Gamble
Rick Schendel	Target
Sylvia Stein	EAN Netherlands
Richard Swan	T3C1
Michael Szafranski	Kraft
Nancy Tai	Georgia Pacific
Ken Traub	Connecterra, Inc.

2098 **URI Representation Group**

Dipan Anarkat	EPCglobal
Michael Mealling	VeriSign
Ken Traub	Connecterra, Inc.

2099

2100 This document also draws from the previous work at the Auto-ID Center, and we
 2101 recognize the contribution of the following individuals: David Brock (MIT), Joe Foley
 2102 (MIT), Sunny Siu (MIT), Sanjay Sarma (MIT), and Dan Engels (MIT). In addition, we
 2103 recognize the contribution from Steve Rehling (P&G) on EPC to GTIN mapping.

2104 The following papers capture the contributions of these individuals:

2105 Engels, D., Foley, J., Waldrop, J., Sarma, S. and Brock, D., "The Networked Physical
 2106 World: An Automated Identification Architecture"
 2107 2nd IEEE Workshop on Internet Applications (WIAPP '01),
 2108 (<http://csdl.computer.org/comp/proceedings/wiapp/2001/1137/00/11370076.pdf>)

2109 Brock, David. "The Electronic Product Code (EPC), A Naming Scheme for Physical
 2110 Objects", 2001. (<http://www.autoidlabs.org/whitepapers/MIT-AUTOID-WH-002.pdf>)

2111 Brock, David. "The Compact Electronic Product Code; A 64-bit Representation of the
 2112 Electronic Product Code", 2001. (<http://www.autoidlabs.com/whitepapers/MIT-AUTOID-WH-008.pdf>)
 2113

2114 **8 References**

2115 [EANUCCGS] "General EAN.UCC Specifications." Version 5.0, EAN International and
 2116 the Uniform Code Council, IncTM, January 2004.

2117 [MIT-TR009] D. Engels, "The Use of the Electronic Product CodeTM," MIT Auto-ID
 2118 Center Technical Report MIT-TR007, February 2003,
 2119 (<http://www.autoidlabs.com/whitepapers/mit-autoid-tr009.pdf>)

2120 [RFC2141] R. Moats, "URN Syntax," Internet Engineering Task Force Request for
 2121 Comments RFC-2141, May 1997, <http://www.ietf.org/rfc/rfc2141.txt>.

2122

2124

2125 **9 Appendix A: Encoding Scheme Summary Tables**

2126

SGTIN Summary						
SGTIN-64	Header	Filter Value	Company Prefix Index		Item Reference	Serial Number
	2 bits	3 bits	14 bits		20 bits	25 bits
	10	8	16,383		9 - 1,048,575	33,554,431
	(Binary value)	(Decimal capacity)	(Decimal capacity)		(Decimal capacity*)	(Decimal capacity)
SGTIN-96	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
	8	3	3	20-40	24 - 4	38
	0011 0000	8	8	999,999 – 999,999,999,999	9,999,999 – 9	274,877,906,943
	(Binary value)	(Decimal capacity)	(Decimal capacity)	(Decimal capacity**)	(Decimal capacity**)	(Decimal capacity)
Filter Values (Non-normative)		SGTIN Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Item Reference and Indicator Digit	
			Bits	Digits	Bits	Digit
Unspecified	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

2127 *Capacity of Item Reference field varies with the length of the Company Prefix

2128 **Capacity 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)	8 (Decimal capacity)	16,383 (Decimal capacity)		99,999 - 99,999,999,999 (Decimal capacity*)	
SSCC-96	Header	Filter Value	Partition	Company Prefix	Serial Reference	Unallocated
	8	3	3	20-40	38-18	24
	0011 0001 (Binary value)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,999,999 (Decimal capacity**)	99,999,999,999 – 99,999 (Decimal capacity**)	[Not Used]
Filter Values (Non-normative)		SSCC Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Serial Reference and extension digit	
Unspecified	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

2130 *Capacity of Serial Reference field varies with the length of the Company Prefix

2131 **Capacity of Company Prefix and Serial Reference fields vary according to the contents of the Partition field.

SGLN Summary						
SGLN-64	Header	Filter Value	Company Prefix Index		Location Reference	Serial Number
	8	3	14		20	19
	0000 1001 (Binary value)	8 (Decimal capacity)	16,383 (Decimal capacity)		999,999 - 0 (Decimal capacity*)	524,288 (Decimal capacity) [Not Used]
SGLN-96	Header	Filter Value	Partition	Company Prefix	Location Reference	Serial Number
	8	3	3	20-40	21-1	41
	0011 0010 (Binary value)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,999,999 (Decimal capacity**)	999,999 – 0 (Decimal capacity**)	2,199,023,255,552 (Decimal capacity) [Not Used]
Filter Values (Non-normative)		SGLN Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Location Reference	
			Bits	Digits	Bits	Digit
Unspecified	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

2133 *Capacity of Location Reference field varies with the length of the Company Prefix

2134 **Capacity of Company Prefix and Location Reference fields vary according to contents of the Partition field.

GRAI Summary						
GRAI-64	Header	Filter Value	Company Prefix Index		Asset Type	Serial Number
	8	3	14		20	19
	0000 1010 (Binary value)	8 (Decimal capacity)	16,383 (Decimal capacity)		999,999 - 0 (Decimal capacity*)	524,288 (Decimal capacity)
GRAI-96	Header	Filter Value	Partition	Company Prefix	Asset Type	Serial Number
	8	3	3	20-40	24 - 4	38
	0011 0011 (Binary value)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,999,999 (Decimal capacity**)	999,999 – 0 (Decimal capacity**)	274,877,906,943 (Decimal capacity)
Filter Values (Non-normative)		GRAI Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Asset Type	
Unspecified	000		Bits	Digits	Bits	Digit
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

2136 *Capacity of Asset Type field varies with Company Prefix.

2137 **Capacity of Company Prefix and Asset Type fields vary according to contents of the Partition field.

GIAI Summary					
GIAI-64	Header	Filter Value	Company Prefix Index		Individual Asset Reference
	8	3	14		39
	0000 1011 (Binary value)	8 (Decimal capacity)	16,383 (Decimal capacity)		549,755,813,888 (Decimal capacity)
GIAI-96	Header	Filter Value	Partition	Company Prefix	Individual Asset Reference
	8	3	3	20-40	62-42
	0011 0100 (Binary value)	8 (Decimal capacity)	8 (Decimal capacity)	999,999 – 999,999,999,999 (Decimal capacity*)	4,611,686,018,427,387,904 - 4,398,046,511,103 (Decimal capacity*)
Filter Values (To be confirmed)		GIAI Partition Table			
Type	Binary Value	Partition Value	Company Prefix		Individual Asset Reference
			Bits	Digits	Bits Digits
Unspecified	000				
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

2139

2140 *Capacity of Company Prefix and Individual Asset Reference fields vary according to contents of the
 2141 Partition field.

2142

2143 **10 Appendix B: EPC Header Values and Tag Identity**
2144 **Lengths**

2145 With regards to tag identity lengths and EPC Header values: In the decoding process of a
2146 single tag: Having knowledge of the identifier length during the signal decoding process
2147 of the reader enables the reader to know when to stop trying to decode bit values.

2148 Knowing when to stop enables the readers to be more efficient in reading speed. For
2149 example, if the same Header value is used at 64 and 96 bits, the reader, upon finding that
2150 header value, must try to decode 96 bits. After decoding 96 bits, the reader must check
2151 the CRC (Cyclic Redundancy Check error check code) against both the 64-bit and 96-bit
2152 numbers it has decoded. If both error checks fail, the numbers are thrown away and the
2153 tag reread. If one of the numbers passes the error check, then that is reported as the valid
2154 number. Note that there is a non-zero, i.e., greater than zero but very small, probability
2155 that an erroneous number can be reported in this process. If both numbers pass the error
2156 check, then there is a problem. Note that there is a small probability that both a 64 bit

2157 EPC and 96-bit EPC whose first 64 bits are the same as the 64-bit EPC will have the
2158 same CRC. Other measures would have to be taken to determine which of the two
2159 numbers is valid (and perhaps both are). All of this slows down the reading process and
2160 introduces potential errors in identified numbers (erroneous numbers may be reported)
2161 and non-identified numbers (tags may be unread due to some of the above). These
2162 problems are primarily evident while reading weakly replying tags, which are often the
2163 tags furthest from the reader antenna and in noisy environments. Encoding the length
2164 within the Header eliminates virtually all of the error probabilities above and those that
2165 remain are reduced significantly in probability.

2166 In the decoding process of multiple tags responding: When multiple tags respond at the
2167 same time their communications will overlap in time. Tags of the same length overlap
2168 almost completely bit for bit when the same reader controls them. Tags of different
2169 lengths will overlap almost completely over the first bits, but the longer tag will continue
2170 communicating after the shorter tag has stopped. Tags of very strong communication
2171 strength will mask tags responding with much weaker strength. The reader can use
2172 communication signal strength as a determiner of when to stop looking to decode bits.
2173 Tags of almost equal communication strength will tend to interfere almost completely
2174 with one another over the first bits before the shorter tag stops. The reader can usually
2175 detect these collisions, but not always when weak signals are trying to be pulled out of
2176 noise, as is the case for the distant tags. When the tags reply with close, but not equal
2177 strength, it may be possible to decode the stronger signal. When the short tag has the
2178 stronger signal, it may be possible to decode the weaker longer tag signal without being
2179 able to definitively say that a second tag is responding due to changes in signal strength.
2180 These problems are primarily evident in weakly replying tags. Encoding the length in the
2181 Header enables the reader to know when to stop pulling out the numbers, which enables it
2182 to more efficiently determine the validity of the numbers.

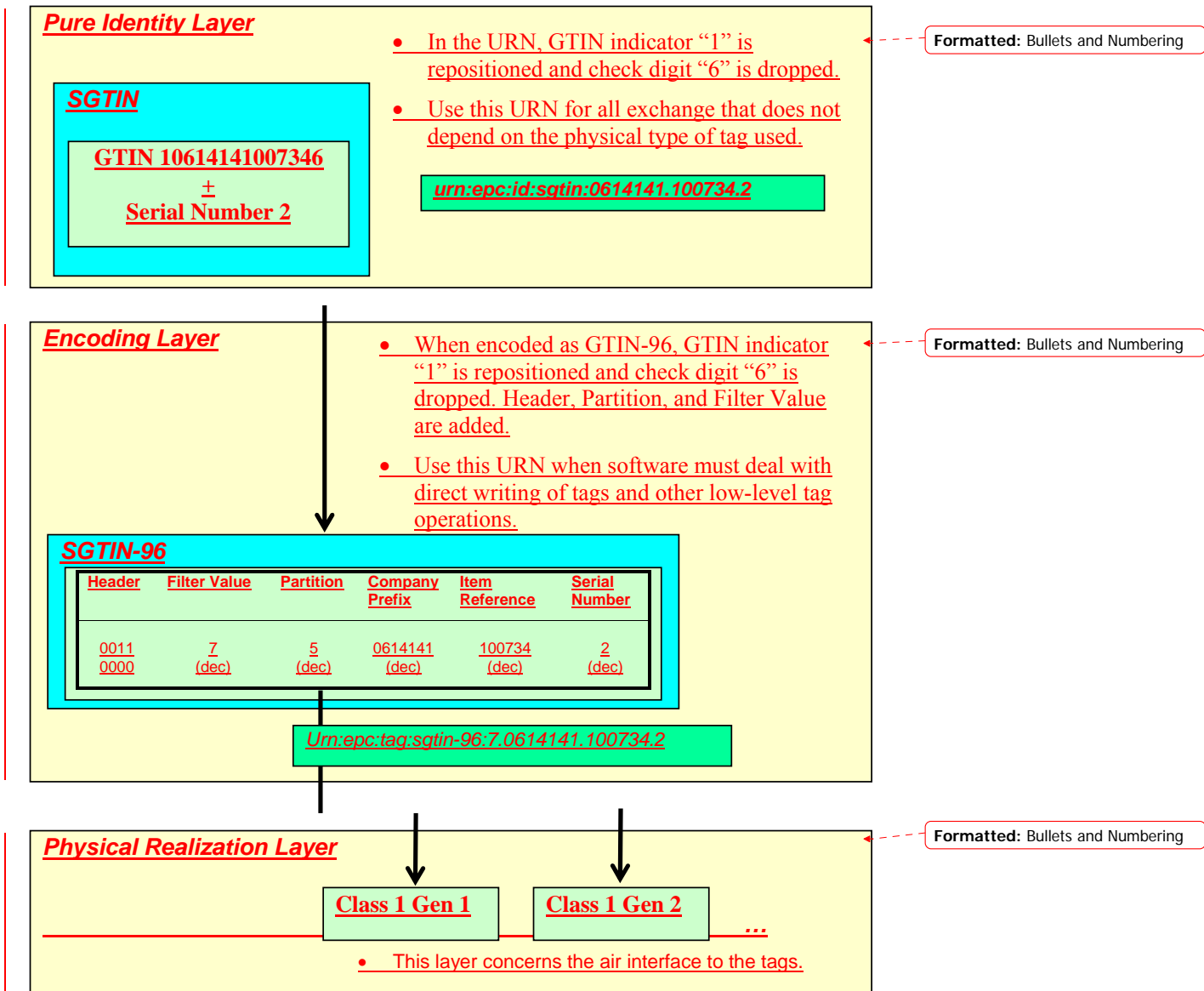
2183 In the identification process: The reader can "select" what length tags it wishes to
2184 communicate with. This eliminates the decoding problems encountered above, since all

2185 communicating tags are of the same length and the reader knows what that length is a
2186 priori. For efficiency reasons, a single selection for a length is preferred, but two can be
2187 workable. More than two becomes very inefficient.
2188 The net effect of encoding the length within the Header is to reduce the probabilities of
2189 error in the decoding process and to increase the efficiency of the identification process.

2190
2191
2192
2193
2194
2195

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

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



2196



2197

2198

2199

2200

	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)	7 (Decimal value)	5 (Decimal value)	0614141 (Decimal value)	100734 (Decimal value)	2 (Decimal value)

2201

2202

2203

2204

- (01) is the Application Identifier for GTIN, and (21) is the Application Identifier for Serial Number. Application Identifiers are used in certain bar codes. The header fulfills this function (and others) in EPC.

2205

- Header for SGTIN-96 is 00110000.

2206

2207

- Filter Value was not defined when this example was created , so 7 is a notional value.

2208

2209

- Since the Company Prefix is seven-digits long (0614141), the Partition value is 5. This means Company Prefix has 24 bits and Item Reference has 20 bits.

2210

- Indicator digit 1 is repositioned as the first digit in the Item Reference.

2211

- Check digit 6 is dropped.

2212

2213

Explanation of SGTIN Filter Values (non-normative).

2214

2215

2216

2217

2218

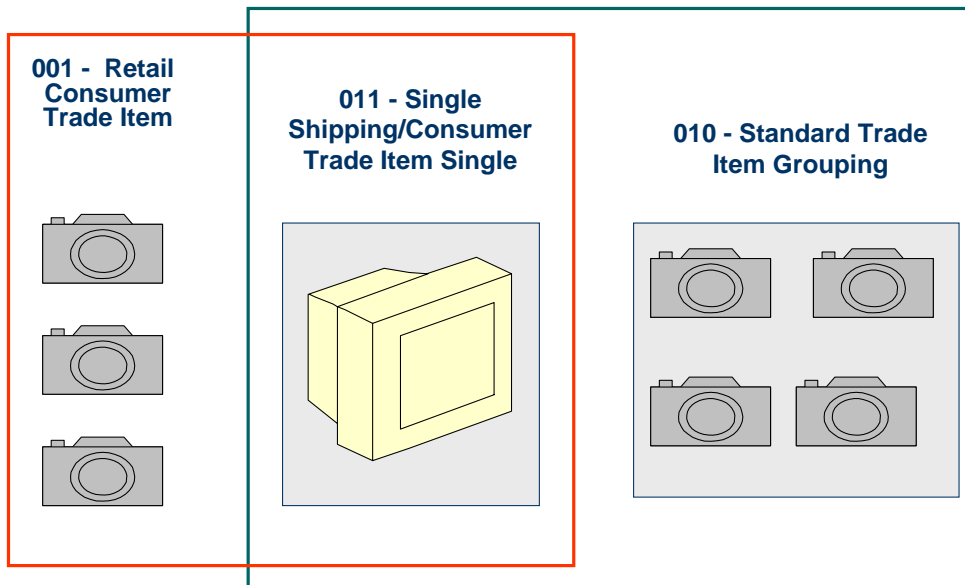
SGTINs can be assigned at several levels, including: item, inner pack, case, and pallet. RFID can read through cardboard, and reading un-needed tags can slow us down, so Filter Values are used to “filter in” desired tags, or “filter out” unwanted tags. Filter values are used within the key type (i.e. 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

2219 minimum of values for now until the community gains more practical experience in their
2220 use. Therefore the three major categories of SGTIN filter values can be thought of in the
2221 following high level terms:

- 2222 • Single Unit: A Retail Consumer Trade Item
- 2223 • Not-a-single unit: A Standard Trade Item Grouping
- 2224 • Items that could be included in both categories: For example, a Single Shipping
2225 container that contains a Single Consumer Trade Item

2226

Three Filter Values



2227

2228

2229 **12 Appendix D: Binary Digit Capacity Tables**

2230

Length in Binary Digits	Decimal Capacity	Length in Binary Digits	Decimal Capacity
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		

2231

2232 **13 Appendix E: List of Abbreviations**

2233

BAG	Business Action Group
EPC	Electronic Product Code
EPCIS	EPC Information Services
GIAI	Global Individual Asset 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

2234

2235

2236 **14 Appendix F: General EAN.UCC Specifications**

2237 (Section 3.0 Definition of Element Strings and Section 3.7 EPCglobal Tag Data
2238 Standard.)

2239 This section provides EAN.UCC approval of this version of the EPCglobal® Tag Data
2240 Standard with the following EAN.UCC Application Identifier definition restrictions:

2241 Companies should use the EAN.UCC specifications to define the applicable fields in
2242 databases and other ICT-systems.

2243 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:

2244 • **AI (00) SSCC** (no restrictions)

2245 • **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is
2246 restricted to permit assignment of 33,554,431 numeric-only serial numbers.

2247 • **AI (41n) GLN + AI (21) Serial Number:** The Tag Data Standard V1.1 R1.23 is approved
2248 with a complete restriction on GLN serialization because this question has not been
2249 resolved by GSMP at this time.

2250 • **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier
2251 definition is restricted to permit assignment of 524,288 numeric-only serial numbers and
2252 the serial number element is mandatory.

2253 • **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier
2254 definition is restricted to permit assignment of 549,755,813,888 numeric-only serial
2255 numbers.

2256 For EAN.UCC use of EPC96-bit tags, the following applies:

2257 • **AI (00) SSCC** (no restrictions)

2258 • **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is
2259 restricted to permit assignment of 274,877,906,943 numeric-only serial numbers)

2260 • **AI (41n) GLN + AI (21) Serial Number:** The Tag Data Standard V1.1 R1.23 is approved
2261 with a complete restriction on GLN serialization because this question has not been
2262 resolved by GSMP at this time.

2263 • **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier
2264 definition is restricted to permit assignment of 274,877,906,943 numeric-only serial
2265 numbers and the serial number element is mandatory.

2266 • **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier
2267 definition is restricted to permit assignment of 4,611,686,018,427,387,904 numeric-only
2268 serial numbers.

2269