



1

2

EPCglobal Tag Data Standards Version 1.3.1

3

4

Specification was Ratified on March 8, 2006

5

The Improvements to correct errata were approved on September 28, 2007

6

7

Disclaimer

8

EPCglobal Inc™ is providing this document as a service to interested industries.

9

This document was developed through a consensus process of interested parties.

10

Although efforts have been to assure that the document is correct, reliable, and

11

technically accurate, EPCglobal Inc. makes NO WARRANTY, EXPRESS OR

12

IMPLIED, THAT THIS DOCUMENT IS CORRECT, WILL NOT REQUIRE

13

MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE,

14

OR WILL BE SUITABLE FOR ANY PURPOSE OR WORKABLE IN ANY

15

APPLICATION, OR OTHERWISE. Use of this document is with the understanding

16

that EPCglobal Inc. has no liability for any claim to the contrary, or for any damage

17

or loss of any kind or nature.

18

19

Copyright notice

20

© 2006, 2007, EPCglobal Inc.

21

All rights reserved. Unauthorized reproduction, modification, and/or use of this document is not

22

permitted. Requests for permission to reproduce should be addressed to

23

epcglobal@epcglobalinc.org.

24

25

EPCglobal Inc.™ is providing this document as a service to interested industries. This document

26

was developed through a consensus process of interested parties. Although efforts have been to

27

assure that the document is correct, reliable, and technically accurate, EPCglobal Inc. makes NO

28

WARRANTY, EXPRESS OR IMPLIED, THAT THIS DOCUMENT IS CORRECT, WILL NOT

29

REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE,

30

OR WILL BE SUITABLE FOR ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR

31

OTHERWISE. Use of this Document is with the understanding that EPCglobal Inc. has no liability

32

for any claim to the contrary, or for any damage or loss of any kind or nature

33

DOCUMENT HISTORY

34

Document Number:	
Document Version:	1.3
Document Date :	2005-11-21

35

36

37

38

Document Summary

39

Document Title:	EPC™ Tag Data Standards Version 1.3
Owner:	Tag Data Standard Work Group
Status:	(<i>check one box</i>) <input type="checkbox"/> DRAFT <input checked="" type="checkbox"/> Approved

40

41

Document Change History

42

Date of Change	Version	Reason for Change	Summary of Change
9/19/2007	1.3.1	Editorial Changes	• GRAI-170, GIAI-202,SGLN-195, GRAI-96
			•
			•
			•
			•

43

44 **Abstract**

45 This document defines the EPC Tag Data Standards version 1.3. It applies to RFID tags
46 conforming to “EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID
47 Protocol for Communications at 860 MHz-960MHz Version 1.0.9” (“Gen2 Specification”).
48 Such tags will be referred to as “Gen 2 Tags” in the remainder of this document. These
49 standards define completely that portion of EPC tag data that is standardized, including how
50 that data is encoded on the EPC tag itself (i.e. the EPC Tag Encodings), as well as how it is
51 encoded for use in the information systems layers of the EPC Systems Network (i.e. the EPC
52 URI or Uniform Resource Identifier Encodings).

53
54 The EPC Tag Encodings include a Header field followed by one or more Value Fields. The
55 Header field defines the overall length and format of the Values Fields. The Value Fields
56 contain a unique EPC Identifier and a required Filter Value when the latter is judged to be
57 important to encode on the tag itself.

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

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

70

71 **Status of this document**

72 This section describes the status of this document at the time of its publication. Other
73 documents may supersede this document. The latest status of this document series is
74 maintained at EPCglobal. This document is based on the Ratified Specification named Tag
75 Data Standards Version 1.3 as ratified by the EPCglobal Board of Governors on March 8,
76 2006. This version corrects identified errata found in the version 1.3 and is marked as
77 version 1.3.1. Comments on this document should be sent to epcinfo@epcglobalinc.org.

78

79 **Changes from Previous Versions**

80 Version 1.3.1

81

82 This update to the Tag Data Standards provides errata changes found since Version 1.3 was
83 published. Changes are as follows

84

- 85 1. In section 3.8.2.2 GRAI-170 Decoding Procedure, the bit numbering has been
86 corrected. For instance “00110111 $b_{162}b_{161}\dots b_0$ “ has been corrected to read
87 “00110111 $b_{161}b_{160}\dots b_0$ “ and so forth throughout the section..
- 88 2. The GIAI-202 Table 23 and the Associated Summary Table in Appendix A did not
89 add up to a total of 188 bits for each Company Prefix/Individual Asset Reference
90 which is what the encoding/decoding procedure expects. The Individual Asset
91 Reference Bits column has been changed so each row adds to 188 bits. For example,
92 for Partition value 0 the Individual Asset Reference bits value “126” was changed to
93 “148”.
- 94 3. An addition error in the Appendix B table, SGLN-195 row, has been corrected.
95 TheTotal bits required column was changed from 333 to 336.
- 96 4. A typographical error in line three of the section 3.8.1.1 GRAI-96 Encoding
97 Procedure has been corrected. The formula “ $15 \leq K \leq 0$ ” was replaced with
98 “ $15 \leq K \leq 30$ ”.
- 99 5. In Section 5.4 (Gen 2 Tag EPC Memory into Tag or Raw URI) step 8 line 4 a
100 missing dot (.) character after the value of A has been corrected.
- 101 6. The arrows in Appendix C between the Bar Code symbol and the SGTIN-96 have
102 been adjusted to reflect the connections between the Company Prefix, Item Reference
103 and Serial Number.

104

105 Version 1.3

106

107 This Tag Data Standards Version 1.3 is aimed for use in Gen 2 Tags, whereas the previous
108 Version 1.1, was aimed for use in UHF Class 1 Generation 1 tags. Version 1.3 maintains
109 compatibility with version 1.1 in the identity level. In other words, this version will continue
110 to support the EAN.UCC system and DoD identity types.

111 However, in Version 1.3, there are significant changes to prior versions, including:

- 112 1. The deprecation of 64 bit encodings.
- 113 2. The elimination of tiered header rules.
- 114 3. The encoding of EPC to fit the structure of Gen 2 Tags
- 115 4. The addition of the Extension Component to the SGLN

- 116 5. Addition of SGTIN-198, SGLN-195, GRAI-170, GIAI-202 and corresponding
117 changes in URI expression for alpha-numeric serial number encoding.
118

119 **Table of Contents**

120 1 Introduction 9

121 2 Identity Concepts..... 10

122 2.1 Pure Identities 11

123 2.1.1 General Types 11

124 2.1.2 EAN.UCC System Identity Types 12

125 2.1.2.1 Serialized Global Trade Item Number (SGTIN)..... 13

126 2.1.2.2 Serial Shipping Container Code (SSCC) 14

127 2.1.2.3 Serialized Global Location Number (SGLN)..... 15

128 2.1.2.4 Global Returnable Asset Identifier (GRAI) 17

129 2.1.2.5 Global Individual Asset Identifier (GIAI)..... 17

130 2.1.3 DoD Identity Type 18

131 3 EPC Tag Bit-level Encodings 18

132 3.1 Headers 19

133 3.2 Use of EPCs on UHF Class 1 Generation 2 Tags..... 21

134 3.2.1 EPC Memory Contents 21

135 3.2.2 The Length Bits..... 23

136 3.3 Notational Conventions 23

137 3.4 General Identifier (GID-96)..... 25

138 3.4.1.1 GID-96 Encoding Procedure 25

139 3.4.1.2 GID-96 Decoding Procedure..... 26

140 3.5 Serialized Global Trade Item Number (SGTIN) 26

141 3.5.1 SGTIN-96 26

142 3.5.1.1 SGTIN-96 Encoding Procedure 29

143 3.5.1.2 SGTIN-96 Decoding Procedure 30

144 3.5.2 SGTIN-198 30

145 3.5.2.1 SGTIN-198 Encoding Procedure 32

146 3.5.2.2 SGTIN-198 Decoding Procedure 32

147 3.6 Serial Shipping Container Code (SSCC)..... 33

148 3.6.1 SSCC-96 33

149 3.6.1.1 SSCC-96 Encoding Procedure 35

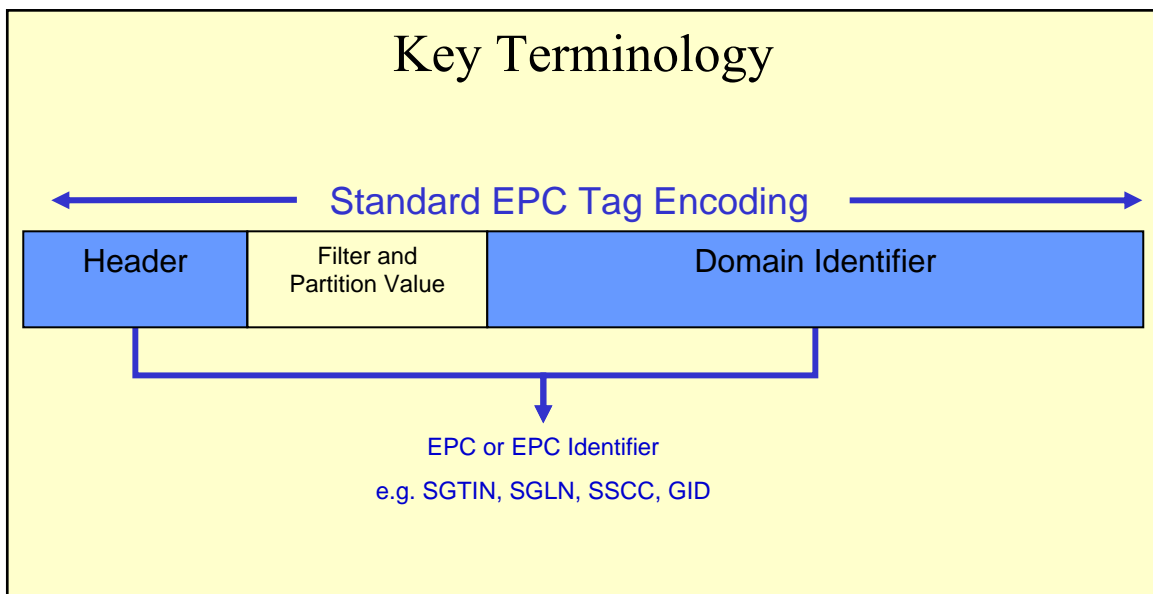
150	3.6.1.2	SSCC-96 Decoding Procedure	36
151	3.7	Serialized Global Location Number (SGLN)	37
152	3.7.1	SGLN-96	37
153	3.7.1.1	SGLN-96 Encoding Procedure	39
154	3.7.1.2	SGLN-96 Decoding Procedure	40
155	3.7.2	SGLN-195	41
156	3.7.2.1	SGLN-195 Encoding Procedure	42
157	3.7.2.2	SGLN-195 Decoding Procedure	43
158	3.8	Global Returnable Asset Identifier (GRAI)	44
159	3.8.1	GRAI-96	44
160	3.8.1.1	GRAI-96 Encoding Procedure	45
161	3.8.1.2	GRAI-96 Decoding Procedure	46
162	3.8.2	GRAI-170	47
163	3.8.2.1	GRAI-170 Encoding Procedure	48
164	3.8.2.2	GRAI-170 Decoding Procedure	48
165	3.9	Global Individual Asset Identifier (GIAI)	49
166	3.9.1	GIAI-96	50
167	3.9.1.1	GIAI-96 Encoding Procedure	51
168	3.9.1.2	GIAI-96 Decoding Procedure	52
169	3.9.2	GIAI-202	53
170	3.9.2.1	GIAI-202 Encoding Procedure	54
171	3.9.2.2	GIAI-202 Decoding Procedure	55
172	3.10	DoD Tag Data Constructs (non-normative)	56
173	3.10.1	DoD-96	56
174	4	URI Representation	56
175	4.1	URI Forms for Pure Identities	57
176	4.2	URI Forms for Related Data Types	59
177	4.2.1	URIs for EPC Tags	59
178	4.2.2	URIs for Raw Bit Strings Arising From Invalid Tags	60
179	4.2.2.1	Use of the Raw URI with Gen 2 Tags	61
180	4.2.2.2	The Length Field of a Raw URI when using Gen 2 Tags (non-normative)	62
181	4.2.3	URIs for EPC Patterns	62

182	4.2.4	URIs for EPC Pure Identity Patterns	63
183	4.3	Syntax	64
184	4.3.1	Common Grammar Elements	64
185	4.3.2	EPCGID-URI	65
186	4.3.3	SGTIN-URI.....	65
187	4.3.4	SSCC-URI.....	65
188	4.3.5	SGLN-URI.....	65
189	4.3.6	GRAI-URI.....	65
190	4.3.7	GIAI-URI.....	66
191	4.3.8	EPC Tag URI	66
192	4.3.9	Raw Tag URI	67
193	4.3.10	EPC Pattern URI.....	67
194	4.3.11	EPC Identity Pattern URI	68
195	4.3.12	DoD Construct URI.....	69
196	4.3.13	Summary (non-normative)	69
197	5	Translation between EPC-URI and Other EPC Representations	72
198	5.1	Bit string into EPC-URI (pure identity)	73
199	5.2	Bit String into Tag or Raw URI.....	75
200	5.3	Gen 2 Tag EPC Memory into EPC-URI (pure identity)	77
201	5.4	Gen 2 Tag EPC Memory into Tag or Raw URI	78
202	5.5	URI into Bit String	78
203	5.6	URI into Gen 2 Tag EPC Memory	82
204	6	Semantics of EPC Pattern URIs	82
205	7	Background Information	83
206	8	References	84
207	9	Appendix A: Encoding Scheme Summary Tables.....	85
208	10	Appendix B: TDS 1.3 EAN.UCC Identities Bit Allocation and Required Physical Tag	
209		Bit Length for Encoding	90
210	11	Appendix C: Example of a Specific Trade Item (SGTIN)	92
211	12	Appendix D: Decimal values of powers of 2 Table.....	95
212	13	Appendix E: List of Abbreviations.....	96
213	14	Appendix F: General EAN.UCC Specifications.....	97
214	15	Appendix G: EAN.UCC Alphanumeric Character Set.....	98

215 Introduction

216 The Electronic Product Code™ (EPC™) is an identification scheme for universally
217 identifying physical objects via Radio Frequency Identification (RFID) tags and other means.
218 The standardized EPC Tag Encodings consists of an EPC (or EPC Identifier) that uniquely
219 identifies an individual object, as well as a Filter Value when judged to be necessary to
220 enable effective and efficient reading of the EPC tags.

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



228

229 **Figure A.** EPC Terminology

230

231 In this version of the EPC – EPC Version 1.3 – the specific coding schemes include a
232 General Identifier (GID), a serialized version of the EAN.UCC Global Trade Item Number
233 (GTIN®), the EAN.UCC Serial Shipping Container Code (SSCC®), the EAN.UCC Global
234 Location Number (GLN®), the EAN.UCC Global Returnable Asset Identifier (GRAI®), the
235 EAN.UCC Global Individual Asset Identifier (GIAI®) and the DOD Construct.

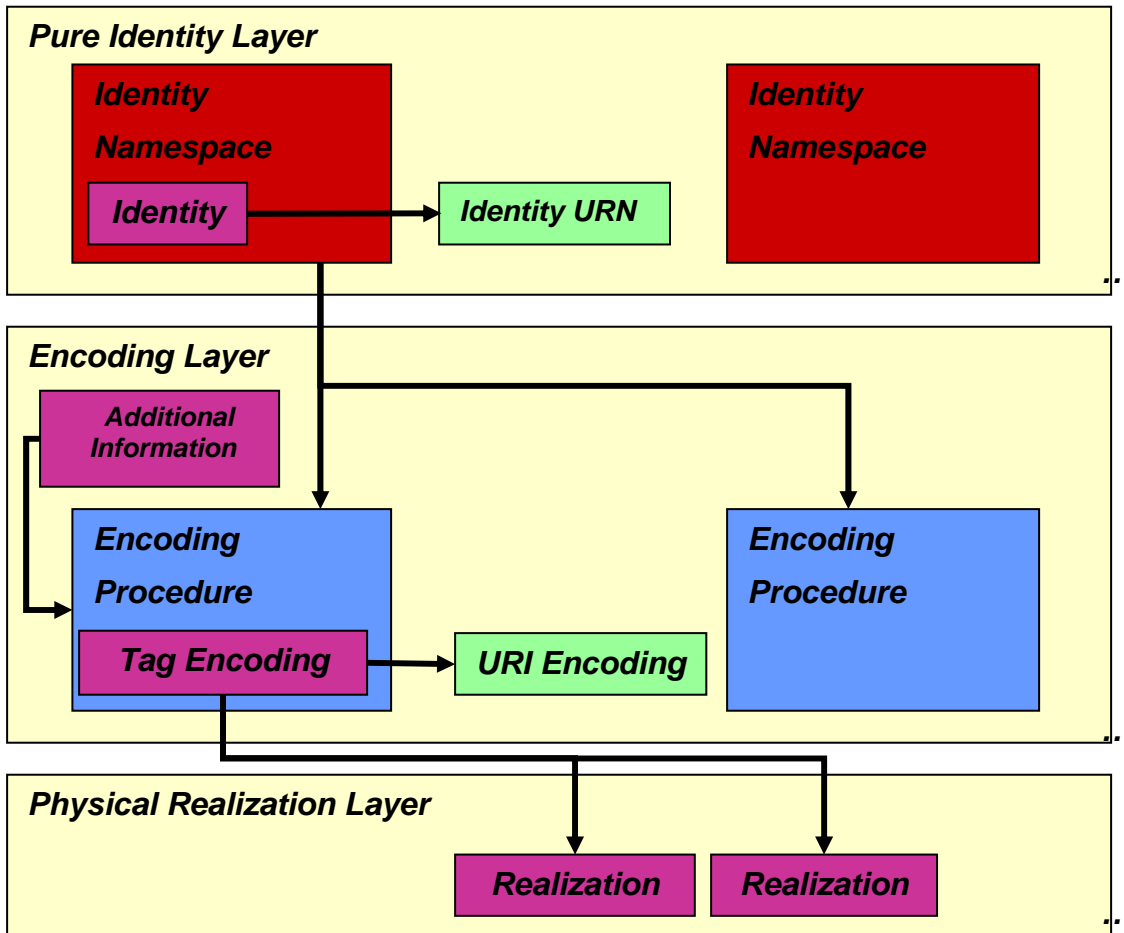
236 In the following sections, we will describe the structure and organization of the EPC and
237 provide illustrations to show its recommended use.

238 The EPCglobal Tag Data Standard V1.3 has been approved by GS1 with the restrictions
239 outlined in the General EAN.UCC Specifications Section 3.7, which is excerpted into Tag
240 Data Standard Appendix F.

241 The latest version of this specification can be [obtained](http://www.epcglobalinc.org/standards/tds/) from EPCglobal at
 242 <http://www.epcglobalinc.org/standards/tds/>

243 1 Identity Concepts

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



248

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

- 250 • Pure identity -- the identity associated with a specific physical or logical entity,
 251 independent of any particular encoding vehicle such as an RF tag, bar code or database
 252 field. As such, a pure identity is an abstract name or number used to identify an entity.
 253 A pure identity consists of the information required to uniquely identify a specific
 254 entity, and no more.
- 255 • Identity URI -- a representation of a pure identity as a Uniform Resource Identifier
 256 (URI). A URI is a character string representation that is commonly used to exchange
 257 identity data between software components of a larger system.

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

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

269 For example, the Serial Shipping Container Code (SSCC) format as defined by the
270 EAN.UCC System is an example of a pure identity. An SSCC encoded into the EPC-SSCC
271 96-bit format is an example of an encoding. That 96-bit encoding, written onto a UHF Class
272 1 RF Tag, is an example of a physical realization.

273 A particular encoding scheme may implicitly impose constraints on the range of identities
274 that may be represented using that encoding. In general, each encoding scheme specifies
275 what constraints it imposes on the range of identities it can represent.

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

283 **1.1 Pure Identities**

284 This section defines the pure identity types for which this document specifies encoding
285 schemes.

286 **1.1.1 General Types**

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

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

296 The *Object Class* is used by an EPC managing entity to identify a class or “type” of thing.
297 These object class numbers, of course, must be unique within each General Manager

298 Number domain. Examples of Object Classes could include case Stock Keeping Units of
299 consumer-packaged goods or different structures in a highway system, like road signs,
300 lighting poles, and bridges, where the managing entity is a County.

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

304 **1.1.2 EAN.UCC System Identity Types**

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

307 The rules regarding the usage of the EAN.UCC codes can be found in the General
308 Specifications of EAN.UCC. This document only explains the incorporation of these
309 numbers in EPC tags.

310 EAN.UCC System codes have a common structure, consisting of a fixed number of decimal
311 digits that encode the identity, plus one additional “check digit” which is computed
312 algorithmically from the other digits. Within the non-check digits, there is an implicit
313 division into two fields: a Company Prefix assigned by GS1 to a managing entity, and the
314 remaining digits, which are assigned by the managing entity. (The digits apart from the
315 Company Prefix are called by a different name by each of the EAN.UCC System codes.)
316 The number of decimal digits in the Company Prefix varies from 6 to 12 depending on the
317 particular Company Prefix assigned. The number of remaining digits therefore varies
318 inversely so that the total number of digits is fixed for a particular EAN.UCC System code
319 type.

320 The GS1 recommendations for the encoding of EAN.UCC System identities into bar codes,
321 as well as for their use within associated data processing software, stipulate that the digits
322 comprising a EAN.UCC System code should always be processed together as a unit, and not
323 parsed into individual fields. This recommendation, however, is not appropriate within the
324 EPC Network, as the ability to divide a code into the part assigned to the managing entity
325 (the Company Prefix in EAN.UCC System types) versus the part that is managed by the
326 managing entity (the remainder) is essential to the proper functioning of the Object Name
327 Service (ONS). In addition, the ability to distinguish the Company Prefix is believed to be
328 useful in filtering or otherwise securing access to EPC-derived data. Hence, the EPC Tag
329 Encodings for EAN.UCC code types specified herein deviate from the aforementioned
330 recommendations in the following ways:

- 331 • EPC Tag Encodings carry an explicit division between the Company Prefix and the
332 remaining digits, with each individually encoded into binary. Hence, converting from
333 the traditional decimal representation of an EAN.UCC System code and an EPC Tag
334 Encoding requires independent knowledge of the length of the Company Prefix.
- 335 • EPC Tag Encodings do not include the check digit. Hence, converting from an EPC Tag
336 Encoding to a traditional decimal representation of a code requires that the check digit
337 be recalculated from the other digits.

338 **1.1.2.1 Serialized Global Trade Item Number (SGTIN)**

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

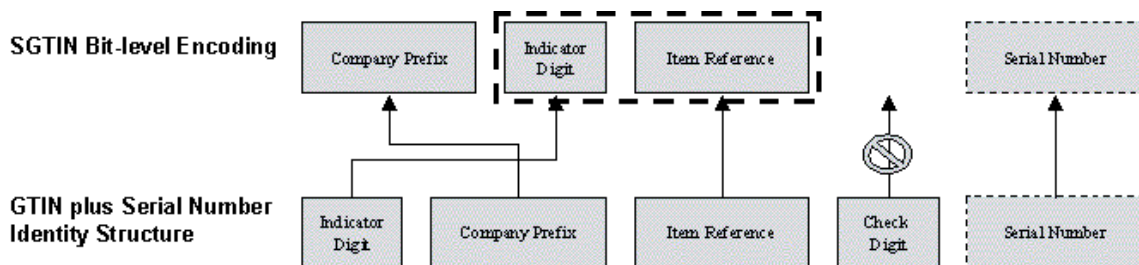
344 *All representations of SGTIN support the full 14-digit GTIN format. This means that the zero*
345 *indicator-digit and leading zero in the Company Prefix for UCC-12, and the zero indicator-*
346 *digit for EAN.UCC-13, can be encoded and interpreted accurately from an EPC Tag*
347 *Encoding. EAN.UCC-8 is not currently supported in EPC, but would be supported in full 14-*
348 *digit GTIN format as well.*

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

353 The SGTIN consists of the following information elements:

- 354 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the
355 same as the Company Prefix digits within an EAN.UCC GTIN decimal code.
- 356 • The *Item Reference*, assigned by the managing entity to a particular object class. The
357 Item Reference for the purposes of EPC Tag Encoding is derived from the GTIN by
358 concatenating the Indicator Digit of the GTIN and the Item Reference digits, and
359 treating the result as a single integer.
- 360 • The *Serial Number*, assigned by the managing entity to an individual object. The serial
361 number is not part of the GTIN code, but is formally a part of the SGTIN.

362



363

364

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

367 The SGTIN is not explicitly defined in the EAN.UCC General Specifications. However, it
368 may be considered equivalent to a EAN.UCC-128 bar code that contains both a GTIN
369 (Application Identifier 01) and a serial number (Application Identifier 21). Serial numbers in
370 AI 21 consist of one to twenty characters, where each character can be a digit, uppercase or
371 lowercase letter, or one of a number of allowed punctuation characters. The complete set of

372 characters allowed is illustrated in Appendix G. The complete AI 21 syntax is supported by
373 the pure identity URI syntax specified in Section 4.3.1.

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

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

383 For the requirement of using longer serial number, or alphabet and other non numeric
384 codings allowed in Application Identifier 21, this version of specification introduces a longer
385 bit encoding format SGTIN-198.

386 *Explanation (non-normative): The restrictions are necessary for 96-bit tags in order for*
387 *serial numbers to fit within the small number of bits available in earlier Class 1 Generation*
388 *1 tags. The serial number range is restricted to numeric values and alphanumeric serial*
389 *numbers are disallowed. Leading zeros are forbidden so that the serial number can be*
390 *considered as a decimal integer when encoding the integer value in binary. By considering*
391 *it to be a decimal integer, "00034", "034", or "34" (for example) can't be distinguished as*
392 *different integer values. In order to insure that every encoded value can be decoded*
393 *uniquely, serial numbers can't have leading zeros. Then, when the bits*
394 *00000000000000000000000010010 on the tag are seen, the serial number as "34" (not "034" or*
395 *"00034") is decoded.*

396 1.1.2.2 Serial Shipping Container Code (SSCC)

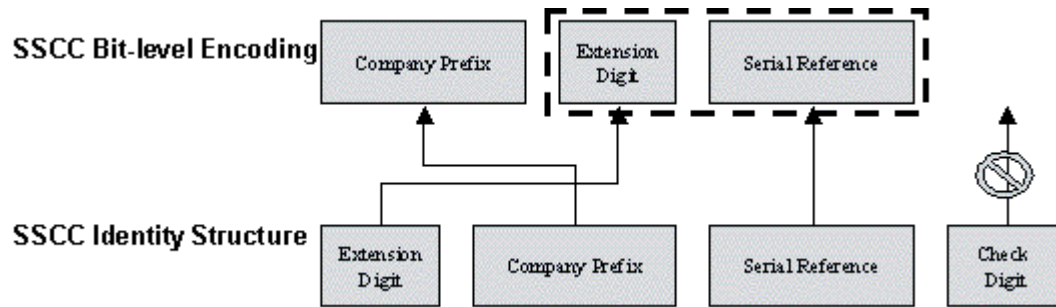
397 The Serial Shipping Container Code (SSCC) is defined by the General EAN.UCC
398 Specifications. Unlike the GTIN, the SSCC is already intended for assignment to individual
399 objects and therefore does not require any additional fields to serve as an EPC pure identity.

400 *Note (Non-Normative): Many applications of SSCC have historically included the*
401 *Application Identifier (00) in the SSCC identifier field when stored in a database. This is not*
402 *a standard requirement, but a widespread practice. The Application Identifier is a sort of*
403 *header used in bar code applications, and can be inferred directly from EPC headers*
404 *representing SSCC. In other words, an SSCC EPC can be interpreted as needed to include*
405 *the (00) as part of the SSCC identifier or not.*

406 The SSCC consists of the following information elements:

- 407 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the
408 same as the Company Prefix digits within an EAN.UCC SSCC decimal code.
- 409 • The *Serial Reference*, assigned uniquely by the managing entity to a specific shipping
410 unit. The Serial Reference for the purposes of EPC Tag Encoding is derived from the
411 SSCC by concatenating the Extension Digit of the SSCC and the Serial Reference
412 digits, and treating the result as a single integer.

413



414

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

416 1.1.2.3 Serialized Global Location Number (SGLN)

417 The Global Location Number (GLN) is defined by the General EAN.UCC Specifications as
418 an identifier of physical or legal entities.

419 A GLN can represent either a discrete, unique physical location such as a dock door or a
420 warehouse slot, or an aggregate physical location such as an entire warehouse. In addition, a
421 GLN can represent a logical entity such as an “organization” that performs a business
422 function such as placing an order.

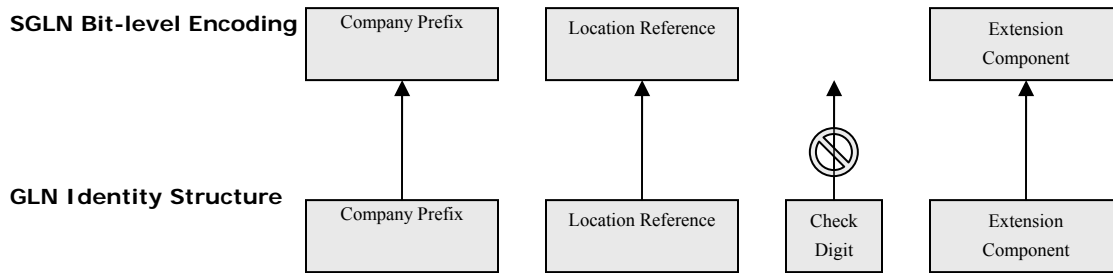
423 Within the GS1 system, high capacity data carriers use Application Identifiers (AI) to
424 distinguish data elements encoded within a single data carrier. The GLN can be associated
425 with many AI’s including physical location, ship to location, invoice to location etc.

426 Recognizing these variables, the EPC SGLN (serialized GLN) represents only the physical
427 location sub-type of GLN AI (414). The serial component is represented by the GLN
428 Extension AI (254). Rules regarding the allocation of a SGLN can be found within the
429 EAN.UCC General Specifications.

430 The SGLN consists of the following information elements:

- 431 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the
432 same as the Company Prefix digits within an EAN.UCC GLN decimal code.
- 433 • The *Location Reference*, assigned uniquely by the managing entity to an aggregate or
434 specific physical location.
- 435 • The *GLN Extension*, assigned by the managing entity to an individual unique location.
 - 436 ➤ The use of the GLN Extension is intended for internal purposes. For communication
437 between trading partners a GLN will be used. The rules defining the use of the
438 SGLN are described in Section 3.7.

439



440

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

442 The SGLN is not explicitly defined in the EAN.UCC General Specifications. However, it
 443 may be considered equivalent to a EAN.UCC-128 bar code that contains both a GLN
 444 (Application Identifier 414) and an Extension Component (Application Identifier 254).
 445 Extension Components in AI 254 consist of one to twenty characters, where each character
 446 can be a digit, uppercase or lowercase letter, or one of a number of allowed punctuation
 447 characters. The complete set of characters allowed is illustrated in Appendix G. The
 448 complete AI 254 syntax is supported by the pure identity URI syntax specified in
 449 Section 4.3.1.

450 When representing Extension Components in 96-bit tags, however, only a subset of the
 451 Extension Component allowed in the General EAN.UCC Specifications for Application
 452 Identifier 254 is permitted. Specifically, the permitted Extension Component are those
 453 consisting of one or more digits characters, with no leading zeros, and whose value when
 454 considered as an integer fits within the range restrictions of the 96-bit tag encodings.

455 While these limitations exist for 96-bit tag encodings, future tag encodings allow a wider
 456 range of Extension Component. Therefore, application authors and database designers
 457 should take the EAN.UCC specifications for Application Identifier 254 into account in order
 458 to accommodate further expansions of the Tag Data Standard.

459 For the requirement of using a longer Extension Component, or alphabet and other non
 460 numeric codings allowed in Application Identifier 254, this version of specification
 461 introduces a longer bit encoding format SGLN-195.

462 *Explanation (non-normative): The restrictions are necessary for 96-bit tags in order for the*
 463 *Extension Component to fit within the small number of bits available in earlier Class 1*
 464 *Generation 1 tags. The Extension Component range is restricted to numeric values and an*
 465 *alphanumeric Extension Component is disallowed. Leading zeros are forbidden so that the*
 466 *Extension Component can be considered as a decimal integer when encoding the integer*
 467 *value in binary. By considering it to be a decimal integer, "00034", "034", or "34" (for*
 468 *example) can't be distinguished as different integer values. In order to insure that every*
 469 *encoded value can be decoded uniquely, Extension Components can't have leading zeros.*
 470 *Then, when the bits 0000000000000000000010010 occurs on the tag, the Extension*
 471 *Component as "34" (not "034" or "00034") is decoded.*

472 .

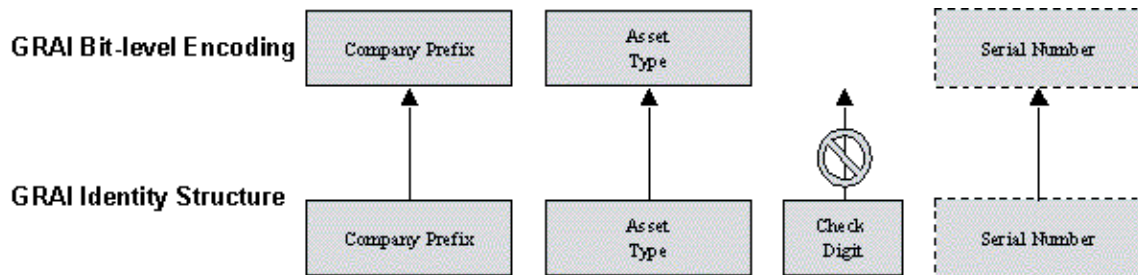
473 **1.1.2.4 Global Returnable Asset Identifier (GRAI)**

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

477

478 The GRAI consists of the following information elements:

- 479 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the
480 same as the Company Prefix digits within an EAN.UCC GRAI decimal code.
- 481 • The *Asset Type*, assigned by the managing entity to a particular class of asset.
- 482 • The *Serial Number*, assigned by the managing entity to an individual object. The GRAI-
483 96 representation is only capable of representing a subset of Serial Numbers allowed in
484 the General EAN.UCC Specifications. Specifically, only those Serial Numbers
485 consisting of one or more digits, with no leading zeros, are permitted [see Appendix F
486 for details].
487 For the requirement of using longer serial number, or alphabet and other non numeric
488 codings allowed in Application Identifier 8003, this version of specification introduces
489 longer bit encoding format GRAI-170.



490

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

492 **1.1.2.5 Global Individual Asset Identifier (GIAI)**

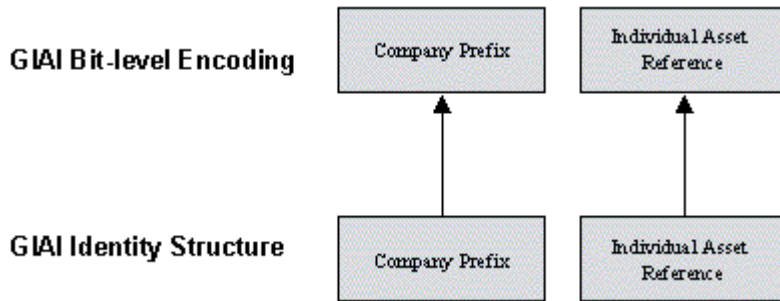
493 The Global Individual Asset Identifier (GIAI) is defined by the General EAN.UCC
494 Specifications. Unlike the GTIN, the GIAI is already intended for assignment to individual
495 objects and therefore does not require any additional fields to serve as an EPC pure identity.

496

497 The GIAI consists of the following information elements:

- 498 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the
499 same as the Company Prefix digits within an EAN.UCC GIAI decimal code.
- 500 • The *Individual Asset Reference*, assigned uniquely by the managing entity to a specific
501 asset. The GIAI-96 representation is only capable of representing a subset of Individual
502 Asset References allowed in the General EAN.UCC Specifications. Specifically, only
503 those Individual Asset References consisting of one or more digits, with no leading
504 zeros, are permitted.
505 For the requirement of using longer serial number, or alphabet and other non numeric

506 codings allowed in Application Identifier 8004, this version of specification introduces
 507 the longer bit encoding format GIAI-202.



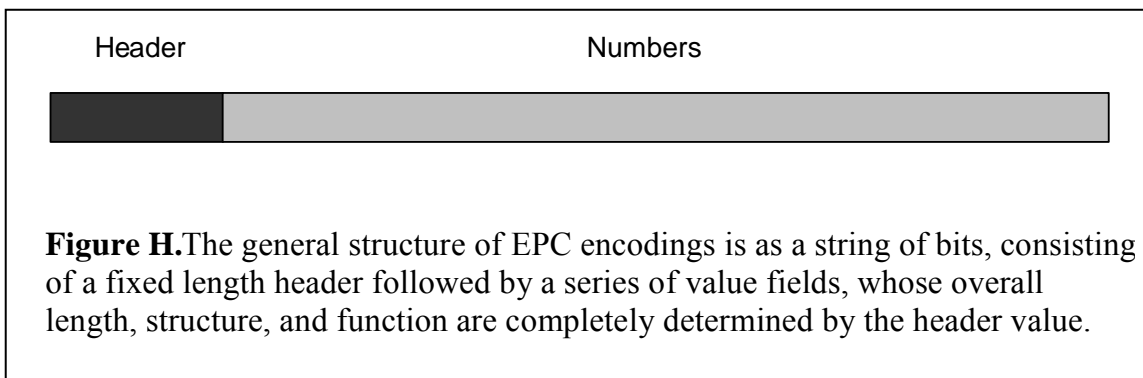
508
 509 **Figure G.** How the parts of the decimal GIAI are extracted and rearranged for encoding.

510 **1.1.3 DoD Identity Type**

511 The DoD Construct identifier is defined by the United States Department of Defense.
 512 This tag data construct may be used to encode 96-bit Class 1 tags for shipping goods to the
 513 United States Department of Defense by a supplier who has already been assigned a CAGE
 514 (Commercial and Government Entity) code.
 515 At the time of this writing, the details of what information to encode into these fields is
 516 explained in a document titled "United States Department of Defense Supplier's Passive
 517 RFID Information Guide" that can be obtained at the United States Department of Defense's
 518 web site (<http://www.dodrfid.org/supplierrguide.htm>).

519 **2 EPC Tag Bit-level Encodings**

520 The general structure of EPC Tag Encodings on a tag is as a string of bits (i.e., a binary
 521 representation), consisting of a fixed length (8-bits) header followed by a series of numeric
 522 fields (Figure H) whose overall length, structure, and function are completely determined by
 523 the header value. For future expansion purpose, a header value of 11111111 is defined, to
 524 indicate that longer header beyond 8-bits is used.



525

526 **2.1 Headers**

527 As previously stated, the Header defines the overall length, identity type, and structure of the
 528 EPC Tag Encoding. Headers defined in this version of the Tag Data Standard are eight bits
 529 in length. The value of 11111111 in the header bits, however, is reserved for future
 530 expansion of header space, so that more than 256 headers may be accommodated by using
 531 longer headers. Therefore, the present specification provides for up to 255 8-bit headers, plus
 532 a currently undetermined number of longer headers.

533 *Back-compatibility note (non-normative) In a prior version of the Tag Data Standard, the*
 534 *header was of variable length, using a tiered approach in which a zero value in each tier*
 535 *indicated that the header was drawn from the next longer tier. For the encodings defined in*
 536 *the earlier specification, headers were either 2 bits or 8 bits. Given that a zero value is*
 537 *reserved to indicate a header in the next longer tier, the 2-bit header had 3 possible values*
 538 *(01, 10, and 11, not 00), and the 8-bit header had 63 possible values (recognizing that the*
 539 *first 2 bits must be 00 and 00000000 is reserved to allow headers that are longer than 8 bits).*
 540 *The 2-bit headers were only used in conjunction with certain 64-bit EPC Tag Encodings.*

541 *In this version of the Tag Data Standard, the tiered header approach has been abandoned.*
 542 *Also, all 64-bit encodings (including all encodings that used 2-bit headers) have been*
 543 *deprecated, and should not be used in new applications. To facilitate an orderly transition,*
 544 *the portions of header space formerly occupied by 64-bit encodings are reserved in this*
 545 *version of the Tag Data Standard, with the intention that they be reclaimed after a “sunset*
 546 *date” has passed. After the “sunset date,” tags containing 64-bit EPCs with 2-bit headers*
 547 *and tags with 64-bit headers starting with 00001 will no longer be properly interpreted.*

548 Eleven encoding schemes have been defined in this version of the EPC Tag Data Standard,
 549 as shown in Table 1 below. The table also indicates header values that are currently
 550 unassigned, as well as header values that have been reserved to allow for an orderly “sunset”
 551 of 64-bit encodings defined in a prior version of the EPC Tag Data Standard. These will not
 552 be available for assignment until after the “sunset date” has passed.

Header Value (binary)	Header Value (hex)	Encoding Length (bits)	Encoding Scheme
0000 0000	00	NA	Unprogrammed Tag
<u>0000 0001</u>	<u>01</u>	NA	Reserved for Future Use
<u>0000 001x</u>	<u>02,03</u>	NA	Reserved for Future Use
<u>0000 01xx</u>	<u>04,05</u>	NA	Reserved for Future Use
	<u>06,07</u>	NA	Reserved for Future Use
0000 1000	08	64	Reserved until 64bit Sunset <SSCC-64>
0000 1001	09	64	Reserved until 64bit Sunset <SGLN-64>
0000 1010	0A	64	Reserved until 64bit Sunset <GRAI-64>
0000 1011	0B	64	Reserved until 64bit Sunset <GIAI-64>

Header Value (binary)	Header Value (hex)	Encoding Length (bits)	Encoding Scheme
<u>0000 1100</u> to <u>0000 1111</u>	0C to 0F		<u>Reserved until 64 bit Sunset</u> <u>Due to 64 bit encoding rule in Gen 1</u>
<u>0001 0000</u> to <u>0010 1110</u>	<u>10</u> to <u>2E</u>	NA NA	<u>Reserved for Future Use</u>
0010 1111	2F	96	DoD-96
0011 0000	30	96	SGTIN-96
0011 0001	31	96	SSCC-96
0011 0010	32	96	SGLN-96
0011 0011	33	96	GRAI-96
0011 0100	34	96	GIAI-96
0011 0101	35	96	GID-96
0011 0110	<u>36</u>	<u>198</u>	<u>SGTIN-198</u>
0011 0111	<u>37</u>	<u>170</u>	<u>GRAI-170</u>
0011 1000	<u>38</u>	<u>202</u>	<u>GIAI-202</u>
0011 1001	<u>39</u>	<u>195</u>	<u>SGLN-195</u>
<u>0011 1010</u> to <u>0011 1111</u>	<u>3A</u> to <u>3F</u>		<u>Reserved for future Header values</u>
0100 0000 to 0111 1111	40 to 7F		<u>Reserved until 64 bit Sunset</u>
1000 0000 to 1011 1111	80 to BF	<u>64</u>	Reserved until 64 bit Sunset <SGTIN-64> (64 header values)

Header Value (binary)	Header Value (hex)	Encoding Length (bits)	Encoding Scheme
<u>1100 0000</u> to <u>1100 1101</u>	<u>C0</u> to <u>CD</u>		Reserved until 64 bit Sunset
1100 1110	CE	64	Reserved until 64 bit Sunset <DoD-64>
<u>1100 1111</u> to <u>1111 1110</u>	CF to FE		<u>Reserved until 64 bit Sunset</u>
1111 1111	FF	NA	Reserved for future headers longer than 8 bits

Table 1. Electronic Product Code Headers

553

554

555 **2.2 Use of EPCs on UHF Class 1 Generation 2 Tags**

556 This section defines how the Electronic Product Code is encoded onto RFID tags conforming
557 to the Gen 2 Specification.

558 In the Gen 2 Specification, the tag memory is separated into four distinct banks, each of
559 which may comprise one or more memory words, where each word is 16 bits long. These
560 memory banks are described as “Reserved”, “EPC”, “TID” and “User”. The “Reserved”
561 memory bank contains kill and access passwords, the “EPC” memory bank contains data
562 used for identifying the object to which the tag is or will be attached, the “TID” memory
563 bank contains data that can be used by the reader to identify the tag’s capability, and “User”
564 memory bank is intended to contain user-specific data.

565 This version of the Tag Data Standards specifies normatively how Electronic Product Codes
566 (EPC) are encoded in the EPC memory bank of Gen 2 Tags. It is anticipated that EPCs may
567 also be used in the User memory bank, but such use is not addressed in this version of the
568 specification. Normative descriptions for encoding of the Reserved and User memory bank
569 will be addressed in future versions of this specification. For encodings of the TID memory
570 bank refer to the Gen 2 Specification.

571 **2.2.1 EPC Memory Contents**

572 The EPC memory bank of a Gen 2 Tag holds an EPC, plus additional control information.
573 The complete contents of the EPC memory bank consist of:

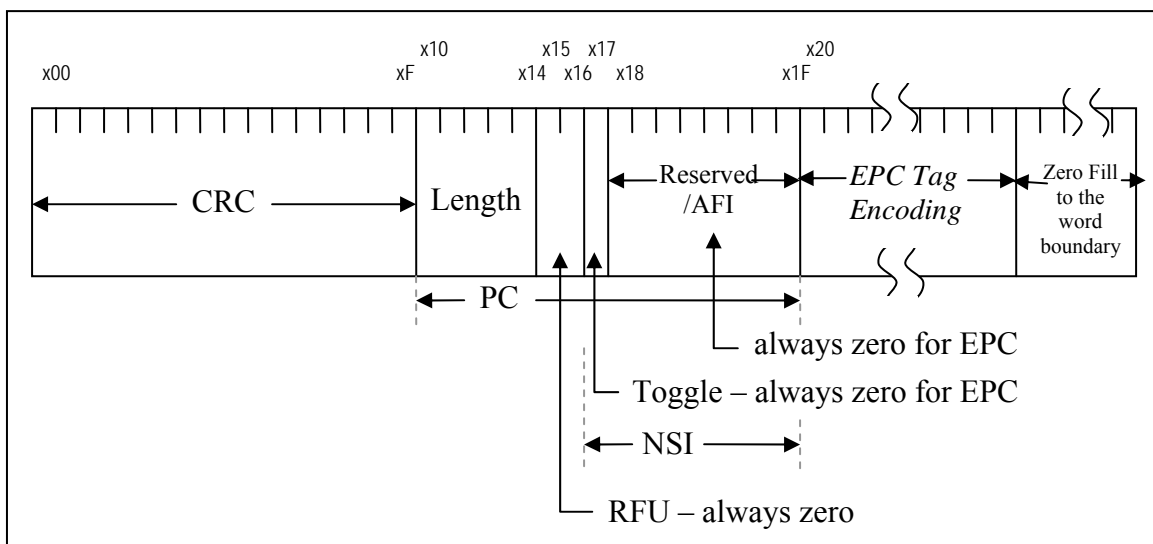
574 • *CRC-16 (16 bits)* Bits that represent the error check code and are auto-calculated by the
575 Tag. (For further details of the CRC, refer to UHF Class 1 Generation 2 Tag Protocol
576 specification Section 6.3.2.1.3)

577 • *Protocol-Control (PC) (16 bits total)* which is subdivided into:

- 578
- 579
- 580
- 581
- 582
- 583
- 584
- 585
- 586
- 587
- 588
- 589
- 590
- 591
- 592
- 593
- 594
- 595
- 596
- 597
- 598
- 599
- *Length (5 bits)* Represents the number of 16-bit words comprising the PC field and the EPC field (below). See discussion below for the encoding of this field.
 - *Reserved for Future Use (RFU) (2 bits)* Always zero in the current version of the UHF Class 1 Generation 2 Tag Protocol Specification.
 - *Numbering System Identifier (NSI) (9 bits total)* which is further subdivided into:
 - *Toggle bit (1 bit)* Boolean flag indicating whether the next 8 bits of the NSI represents reserved memory or an ISO 15961 Application Family Identifier (AFI). If set to “zero” indicates that the NSI contains reserved memory, if set to “one” indicates that the NSI contains an ISO AFI.
 - *Reserved/AFI (8 bits)* Based on the value of the Toggle Bit above, these 8 bits are either Reserved and must all be set to “zero”, or contain an AFI whose value is defined under the authority of ISO.
 - *EPC (variable length)* When the Toggle Bit is set to “zero”, an EPC Tag Encoding as defined in the remaining sections of this chapter is contained here. When the Toggle Bit is set to “one”, these bits are part of a non-EPC coding scheme identified by the AFI field (see above) whose interpretation is outside the scope of this specification.
 - *Zero fill (variable length)* If there is any additional memory beyond EPC Tag Encoding required to meet the 16 bit word boundaries specified in Gen 2 Specification, it is filled with zeros. An implementation shall not put any data into EPC memory following the EPC Tag Encoding and any required zero fill (15 bits or less); if it does, it is not in compliance with the specification and risks the possibility of incompatibility with a future version of the spec.

600

601 The following figure depicts the complete contents of the EPC bank of a Gen 2 Tag,
 602 including the EPC and the surrounding control information, when an EPC is encoded into the
 603 EPC bank:



604

605 **Figure I.** Complete contents of EPC memory bank of a Gen 2 Tag.

606

607 Except for the 16 bit CRC it is the responsibility of the application or process
608 communicating with the reader to provide all the bits to encode in the EPC memory bank.

609 The complete contents of the EPC are defined by the remaining subsections within this
610 chapter.

611 **2.2.2 The Length Bits**

612 The length field is used to let a reader know how much of the EPC memory is occupied with
613 valid data. The value of the length field is the number of 16-bit segments occupied with
614 valid data, not including the CRC, minus one. For example, if set to '000000', the length
615 field indicates that valid data extends through bit x1F, if set to '00001', the length field
616 indicates that valid data extends through bit x2F, and so on.

617 When a Gen 2 Tag contains an EPC Tag Encoding in the EPC bank, the length field is
618 normally set to the smallest number that would contain the particular kind of EPC Tag
619 Encoding in use. Specifically, if the EPC bank contains an N-bit EPC Tag Encoding, then
620 the length field is normally set to $N/16$, rounded up to the nearest integer. For example, with
621 a 96-bit EPC Tag Encoding, the length field is normally set to 6 (00110 in binary).

622 It is important to note that the length of the EPC Tag Encoding is indicated by the EPC
623 header, not by the length field in the PC bits. This is necessarily so, because the length field
624 indicates only the nearest multiple of 16 bits, but the actual amount of EPC memory
625 consumed by the EPC Tag Encoding does not necessarily fall on a multiple-of-16-bit
626 boundary.

627 Moreover, there are applications in which the length field may be set to a different value than
628 the one determined by the formula above. For example, there may be applications in which
629 the EPC is not written to the EPC bank in one operation, but where a prefix of the EPC is
630 written in one operation (perhaps excluding the serial number) and subsequently the
631 remainder of the EPC is written. In such an application, a length field smaller than the
632 normal value might be used to indicate that the EPC is incompletely written.

633

634 **2.3 Notational Conventions**

635 In the remainder of this section, EPC Tag Encoding schemes are depicted using the
636 following notation (See Table 2).

	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
SGTIN-96	8	3	3	20-40	24-4	38
	0011 0000 (Binary value)	(Refer to Table 5 for values)	(Refer to Table 6 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	9,999,999 – 9 (Max. decimal range*)	274,877,906,943 (Max. decimal value)

637

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

638

Table 2. Example of Notation Conventions.

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

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

- 649 • The value of a binary number indicated by (*Binary value*), as is the case for the
650 Header field in the example table above
- 651 • The maximum decimal value indicated by (*Max. decimal value*) of a fixed length
652 field. This is calculated as $2^n - 1$, where n = the fixed number of bits in the field.
- 653 • A range of maximum decimal values indicated by (*Max. decimal range*). This range
654 is calculated using the normative rules expressed in the related encoding procedure
655 section
- 656 • A reference to a table that provides the valid values defined for the field..

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

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

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

668 **2.4 General Identifier (GID-96)**

669 The *General Identifier* is defined for a 96-bit EPC, and is independent of any existing
 670 identity specification or convention. In addition to the header which guarantees uniqueness
 671 in the EPC namespace, the *General Identifier* is composed of three fields - the *General*
 672 *Manager Number*, *Object Class* and *Serial Number*, as shown in Table 3.

673

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

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

676

- 677 • The *Header* is 8-bits, with a binary value of 0011 0101.
- 678 • The *General Manager Number* identifies essentially a company, manager or
 679 organization; that is an entity responsible for maintaining the numbers in subsequent
 680 fields – Object Class and Serial Number. EPCglobal assigns the General Manager
 681 Number to an entity, and ensures that each General Manager Number is unique.

682 *Note (non-normative): Currently, GSI is only allocating an integer value in the range*
 683 *from 95,100,000 to 95,199,999 for this number.*

- 684 • The *Object Class* is used by an EPC managing entity to identify a class or “type” of
 685 thing. These object class numbers, of course, must be unique within each General
 686 Manager Number domain. Examples of Object Classes could include case Stock
 687 Keeping Units of consumer-packaged goods and component parts in an assembly.
- 688 • The *Serial Number* code, or serial number, is unique within each object class. In other
 689 words, the managing entity is responsible for assigning unique – non-repeating serial
 690 numbers for every instance within each object class code.

691 **2.4.1.1 GID-96 Encoding Procedure**

692 The following procedure creates a GID-96 encoding.

693 Given:

- 694 • A General Manager Number M where $0 \leq M < 2^{28}$
- 695 • An Object Class C where $0 \leq C < 2^{24}$
- 696 • A Serial Number S where $0 \leq S < 2^{36}$

697 Procedure:

- 698 1. Construct the General Manager Number by considering digits $d_1d_2\dots d_8$ to be a decimal
699 integer, M . If the value is outside the range specified above, stop: this GID cannot be
700 encoded as a valid GID-96
- 701 2. If the Object class and/or the Serial Number are provided with a value outside the
702 acceptable range specified above, stop: this GID cannot be encoded as a valid GID-96
- 703 3. Construct the final encoding by concatenating the following bit fields, from most
704 significant to least significant: Header 00110101, General Manager Number M (28 bits),
705 Object Class C (24 bits), Serial Number S (36 bits).

706 **2.4.1.2 GID-96 Decoding Procedure**

707 Given:

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

710 Yields:

- 711 • A General Manager Number
- 712 • An Object Class
- 713 • A Serial Number

714 Procedure:

- 715 1. Bits $b_{87}b_{86}\dots b_{60}$, considered as an unsigned integer, are the General Manager Number.
- 716 2. Bits $b_{59}b_{58}\dots b_{36}$, considered as an unsigned integer, are the Object Class.
- 717 3. Bits $b_{35}b_{34}\dots b_0$, considered as an unsigned integer, are the Serial Number.

718 **2.5 Serialized Global Trade Item Number (SGTIN)**

719 The EPC Tag Encoding scheme for SGTIN permits the direct embedding of EAN.UCC
720 System standard GTIN and Serial Number codes on EPC tags. In all cases, the check digit is
721 not encoded.

722

723 **2.5.1 SGTIN-96**

724 In addition to a Header, the SGTIN-96 is composed of five fields: the *Filter Value*, *Partition*,
725 *Company Prefix*, *Item Reference*, and *Serial Number*, as shown in Table 4.

	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
SGTIN-96	8	3	3	20-40	24-4	38
	0011 0000 (Binary value)	(Refer to Table 5 for values)	(Refer to Table 6 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	9,999,999 – 9 (Max. decimal range*)	274,877,906,943 (Max. decimal value)

726
727

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

728

Table 4. The EPC SGTIN-96 bit allocation, header, and maximum decimal values.

729

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

730

- *Filter Value* is not part of the SGTIN pure identity, but is additional data that is used for fast filtering and pre-selection of basic logistics types. The normative specifications for Filter Values are specified in Table 5.

731

732

733

734

735

736

737

738

739

740

741

742

743

The value of 000 means “All Others”. That is, a filter value of 000 means that the object to which the tag is affixed does not match any of the logistic types defined as other filter values in this specification. It should be noted that tags conforming to earlier versions of this specification, in which 000 was the only value approved for use, will have filter value equal to 000, but following the ratification of this standard, the filter value should be set to match the object to which the tag is affixed, and use 000 only if the filter value for such object does not exist in the specification. A Standard Trade Item grouping represents all levels of packaging for logistical units. The Single Shipping / Consumer Trade item type should be used when the individual item is also the logistical unit (e.g. Large screen television, Bicycle).

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

744

Table 5. SGTIN Filter Values .

- 745 • *Partition* is an indication of where the subsequent Company Prefix and Item Reference
746 numbers are divided. This organization matches the structure in the EAN.UCC GTIN
747 in which the Company Prefix added to the Item Reference number (prefixed by the
748 single Indicator Digit) totals 13 digits, yet the Company Prefix may vary from 6 to 12
749 digits and the concatenation of single Indicator Digit and Item Reference from 7 to 1
750 digit(s). The available values of *Partition* and the corresponding sizes of the *Company*
751 *Prefix* and *Item Reference* fields are defined in Table 6.
- 752 • *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.
- 753 • *Item Reference* contains a literal embedding of the GTIN Item Reference number. The
754 Indicator Digit is combined with the Item Reference field in the following manner:
755 Leading zeros on the item reference are significant. Put the Indicator Digit in the
756 leftmost position available within the field. *For instance, 00235 is different than 235.*
757 *With the indicator digit of 1, the combination with 00235 is 100235.* The resulting
758 combination is treated as a single integer, and encoded into binary to form the *Item*
759 *Reference* field.
- 760 • *Serial Number* contains a serial number. The SGTIN-96 encoding is only capable of
761 representing integer-valued serial numbers with limited range. The EAN.UCC
762 specifications permit a broader range of serial numbers. The EAN.UCC-128 barcode
763 symbology provides for a 20-character alphanumeric serial number to be associated
764 with a GTIN using Application Identifier (AI) 21 [EAN.UCCGS]. It is possible to
765 convert between the serial numbers in the SGTIN-96 tag encoding and the serial
766 numbers in AI 21 barcodes under certain conditions. Specifically, such interconversion
767 is possible when the alphanumeric serial number in AI 21 happens to consist only of
768 digits with no leading zeros, and whose value when interpreted as an integer falls
769 within the range limitations of the SGTIN-96 tag encoding. These considerations are
770 reflected in the encoding and decoding procedures below.

771

Partition Value (P)	Company Prefix		Indicator Digit and Item Reference	
	Bits (M)	Digits (L)	Bits (N)	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

Partition Value (<i>P</i>)	Company Prefix		Indicator Digit and Item Reference	
	Bits (<i>M</i>)	Digits (<i>L</i>)	Bits (<i>N</i>)	Digits
6	20	6	24	7

Table 6. SGTIN Partitions.

772

773 **2.5.1.1 SGTIN-96 Encoding Procedure**

774 The following procedure creates an SGTIN-96 encoding.

775 Given:

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

781 Procedure:

- 782 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
- 783 the Partition Table (Table 6) to determine the Partition Value, P , the number of bits M in the
- 784 Company Prefix field, and the number of bits N in the Item Reference and Indicator Digit
- 785 field. If L is not found in any row of Table 6, stop: this GTIN cannot be encoded in an
- 786 SGTIN-96.
- 787 2. Construct the Company Prefix by concatenating digits $d_2d_3\dots d_{(L+1)}$ and considering the
- 788 result to be a decimal integer, C .
- 789 3. Construct the Indicator Digit and Item Reference by concatenating digits
- 790 $d_1d_{(L+2)}d_{(L+3)}\dots d_{13}$ and considering the result to be a decimal integer, I .
- 791 4. When the Serial Number is provided directly as an integer S where $0 \leq S < 2^{38}$, proceed to
- 792 Step 5. Otherwise, when the Serial Number is provided as an EAN.UCC-128 Application
- 793 Identifier 21 consisting of characters $s_1s_2\dots s_K$, construct the Serial Number by concatenating
- 794 digits $s_1s_2\dots s_K$. If any of these characters is not a digit, stop: this Serial Number cannot be
- 795 encoded in the SGTIN-96 encoding. Also, if $K > 1$ and $s_1 = 0$, stop: this Serial Number
- 796 cannot be encoded in the SGTIN-96 encoding (because leading zeros are not permitted
- 797 except in the case where the Serial Number consists of a single zero digit). Otherwise,
- 798 consider the result to be a decimal integer, S . If $S \geq 2^{38}$, stop: this Serial Number cannot be
- 799 encoded in the SGTIN-96 encoding.
- 800 5. Construct the final encoding by concatenating the following bit fields, from most
- 801 significant to least significant: Header 00110000 (8 bits), Filter Value F (3 bits), Partition
- 802 Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Item Reference from
- 803 Step 3 (N bits), Serial Number S from Step 4 (38 bits). Note that $M+N = 44$ bits for all P .

804 **2.5.1.2 SGTIN-96 Decoding Procedure**

805 Given:

- 806 • An SGTIN-96 as a 96-bit bit string $00110000b_{87}b_{86}\dots b_0$ (where the first eight bits
807 00110000 are the header)

808 Yields:

- 809 • An EAN.UCC GTIN-14
810 • A Serial Number
811 • A Filter Value

812 Procedure:

- 813 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.
814 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
815 $P = 7$, stop: this bit string cannot be decoded as an SGTIN-96.
816 3. Look up the Partition Value P in Table 6 to obtain the number of bits M in the Company
817 Prefix and the number of digits L in the Company Prefix.
818 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned integer.
819 If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal SGTIN-
820 96 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$, adding
821 leading zeros as necessary to make up L digits in total.
822 5. Extract the Item Reference and Indicator by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{38}$ as an
823 unsigned integer. If this integer is greater than or equal to $10^{(13-L)}$, stop: the input bit string
824 is not a legal SGTIN-96 encoding. Otherwise, convert this integer to a (13-L)-digit decimal
825 number $i_1i_2\dots i_{(13-L)}$, adding leading zeros as necessary to make (13-L) digits.
826 6. Construct a 13-digit number $d_1d_2\dots d_{13}$ where $d_1 = i_1$ from Step 5, $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$
827 from Step 4, and $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2i_3\dots i_{(13-L)}$ from Step 5.
828 7. Calculate the check digit $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8 +$
829 $d_{10} + d_{12})) \bmod 10$.
830 8. The EAN.UCC GTIN-14 is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{14}$.
831 9. Bits $b_{37}b_{36}\dots b_0$, considered as an unsigned integer, are the Serial Number.
832 10. (Optional) If it is desired to represent the serial number as a EAN.UCC-128 Application
833 Identifier 21, convert the integer from Step 9 to a decimal string with no leading zeros. If the
834 integer in Step 9 is zero, convert it to a string consisting of the single character “0”.

835 **2.5.2 SGTIN-198**

836 In addition to a Header, the SGTIN-198 is composed of five fields: the *Filter Value*,
837 *Partition*, *Company Prefix*, *Item Reference*, and *Serial Number*, as shown in Table 7.

	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
SGTIN-198	8	3	3	20-40	24-4	140
	0011 0110 (Binary value)	(Refer to Table 5 for values)	(Refer to Table 6 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	9,999,999 – 9 (Max. decimal range*)	Up to 20 alphanumeric characters

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

840 **Table 7.** The EPC SGTIN-198 bit allocation, header, and maximum decimal values.

- 841 • *Header* is 8-bits, with a binary value of 0011 0110.
- 842 • *Filter Value* is not part of the GTIN or EPC identifier, but is used for fast filtering and
843 pre-selection of basic logistics types. The Filter Values for 96-bit, and 198-bit GTIN
844 are the same. See Table 5.
- 845 • *Partition* is an indication of where the subsequent Company Prefix and Item Reference
846 numbers are divided. This organization matches the structure in the EAN.UCC GTIN
847 in which the Company Prefix added to the Item Reference number (prefixed by the
848 single Indicator Digit) totals 13 digits, yet the Company Prefix may vary from 6 to 12
849 digits and the Item Reference (including the single Indicator Digit) from 7 to 1 digit(s).
850 The available values of *Partition* and the corresponding sizes of the *Company Prefix*
851 and *Item Reference* fields are defined in Table 6.
- 852 • *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.
- 853 • *Item Reference* contains a literal embedding of the GTIN Item Reference number. The
854 Indicator Digit is combined with the Item Reference field in the following manner:
855 Leading zeros on the item reference are significant. Put the Indicator Digit in the
856 leftmost position available within the field. *For instance, 00235 is different than 235.*
857 *With the indicator digit of 1, the combination with 00235 is 100235.* The resulting
858 combination is treated as a single integer, and encoded into binary to form the *Item*
859 *Reference* field.
- 860 • *Serial Number* contains a serial number. The SGTIN-198 encoding is capable of
861 representing alphanumeric serial numbers of up to 20 characters, permitting the full
862 range of serial numbers available in the EAN.UCC-128 barcode symbology using
863 Application Identifier (AI) 21 [EAN.UCCGS]. See Appendix G for permitted values.

864

865 **2.5.2.1 SGTIN-198 Encoding Procedure**

866 The following procedure creates an SGTIN-198 encoding.

867 Given:

- 868 • An EAN.UCC GTIN-14 consisting of digits $d_1d_2\dots d_{14}$
- 869 • The length L of the Company Prefix portion of the GTIN
- 870 • An EAN.UCC-128 Application Identifier 21 consisting of characters $s_1s_2\dots s_K$, where $K \leq$
871 20.
- 872 • A Filter Value F where $0 \leq F < 8$

873 Procedure:

- 874 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
875 the Partition Table (Table 6) to determine the Partition Value, P , the number of bits M in the
876 Company Prefix field, and the number of bits N in the Item Reference and Indicator Digit
877 field. If L is not found in any row of Table 6, stop: this GTIN cannot be encoded in an
878 SGTIN-198.
- 879 2. Construct the Company Prefix by concatenating digits $d_2d_3\dots d_{(L+1)}$ and considering the
880 result to be a decimal integer, C .
- 881 3. Construct the Indicator Digit and Item Reference by concatenating digits
882 $d_1d_{(L+2)}d_{(L+3)}\dots d_{13}$ and considering the result to be a decimal integer, I .
- 883 4. . Check that each of the characters $s_1s_2\dots s_K$ is one of the 82 characters listed in the table
884 in Appendix G. If this is not the case, stop: this character string cannot be encoded as an
885 SGTIN-198. Otherwise construct the Serial Number by concatenating the 7-bit code, as
886 given in Appendix G, for each of the characters $s_1s_2\dots s_K$, yielding $7K$ bits total. If $K < 20$,
887 concatenate additional zero bits to the right to make a total of 140 bits.
- 888 5. Construct the final encoding by concatenating the following bit fields, from most
889 significant to least significant: Header 00110110 (8 bits), Filter Value F (3 bits), Partition
890 Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Item Reference from
891 Step 3 (N bits), Serial Number from Step 4 (140 bits). Note that $M+N = 44$ bits for all P .

892 **2.5.2.2 SGTIN-198 Decoding Procedure**

893 Given:

- 894 • An SGTIN-198 as a 198-bit bit string $00110110b_{189}b_{188}\dots b_0$ (where the first eight bits
895 00110110 are the header)

896 Yields:

- 897 • An EAN.UCC GTIN-14
- 898 • A Serial Number
- 899 • A Filter Value

900 Procedure:

- 901 1. Bits $b_{189}b_{188}b_{187}$, considered as an unsigned integer, are the Filter Value.
- 902 2. Extract the Partition Value P by considering bits $b_{186}b_{185}b_{184}$ as an unsigned integer. If
903 $P = 7$, stop: this bit string cannot be decoded as an SGTIN-198.
- 904 3. Look up the Partition Value P in Table 6 to obtain the number of bits M in the Company
905 Prefix and the number of digits L in the Company Prefix.
- 906 4. Extract the Company Prefix C by considering bits $b_{183}b_{182}\dots b_{(184-M)}$ as an unsigned
907 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal
908 SGTIN-198 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$,
909 adding leading zeros as necessary to make up L digits in total.
- 910 5. Extract the Item Reference and Indicator by considering bits $b_{(183-M)}b_{(182-M)}\dots b_{140}$ as an
911 unsigned integer. If this integer is greater than or equal to $10^{(13-L)}$, stop: the input bit string
912 is not a legal SGTIN-198 encoding. Otherwise, convert this integer to a $(13-L)$ -digit decimal
913 number $i_1i_2\dots i_{(13-L)}$, adding leading zeros as necessary to make $(13-L)$ digits.
- 914 6. Construct a 13-digit number $d_1d_2\dots d_{13}$ where $d_1 = i_1$ from Step 5, $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$
915 from Step 4, and $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2i_3\dots i_{(13-L)}$ from Step 5.
- 916 7. Calculate the check digit $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8 +$
917 $d_{10} + d_{12})) \bmod 10$.
- 918 8. The EAN.UCC GTIN-14 is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{14}$.
- 919 9. Divide the remaining bits $b_{139}b_{138}\dots b_0$ into 7-bit segments. The result should consist of K
920 non-zero segments followed by $20-K$ zero segments. If this is not the case, stop: this bit
921 string cannot be decoded as an SGTIN-198. Otherwise, look up each of the non-zero 7-bit
922 segments in Appendix G to obtain a corresponding character. If any of the non-zero 7-bit
923 segments has a value that is not in Appendix G, stop: this bit string cannot be decoded as an
924 SGTIN-198. Otherwise, the K characters so obtained, considered as a character string, is the
925 value of the EAN.UCC AI 21.
- 926 10. The EAN.UCC SGTIN-198 is the concatenation of the digits from Steps 6 and 7 and the
927 characters from Step 9. $\therefore d_1d_2\dots d_{14} s_1s_2\dots s_K$
- 928
- 929

930 **2.6 Serial Shipping Container Code (SSCC)**

931 The EPC Tag Encoding scheme for SSCC permits the direct embedding of EAN.UCC
932 System standard SSCC codes on EPC tags. In all cases, the check digit is not encoded.

933 **2.6.1 SSCC-96**

934 In addition to a Header, the EPC SSCC-96 is composed of four fields: the *Filter Value*,
935 *Partition*, *Company Prefix*, and *Serial Reference*, as shown in Table 8.

936
937

*Max. decimal value range of Company Prefix and Serial Reference fields vary according to the contents of the

	Header	Filter Value	Partition	Company Prefix	Serial Reference	Unallocated
SSCC-96	8	3	3	20-40	38-18	24
	0011 0001 (Binary value)	(Refer to Table 9 for values)	(Refer to Table 10 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	99,999,999,999 – 99,999 (Max. decimal range*)	[Not Used]

938

Partition field.

939

Table 8. The EPC 96-bit SSCC bit allocation, header, and maximum decimal values.

940

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

941

- *Filter Value* is not part of the SSCC or EPC identifier, but is used for fast filtering and pre-selection of basic logistics types. The normative specifications for Filter Values are specified in Table 9.

942

943

944

945

946

947

948

949

950

The value of 000 means “All Others”. That is, a filter value of 000 means that the object to which the tag is affixed does not match any of the logistic types defined as other filter values in the specification. It should be noted that tags conforming to earlier versions of this specification, in which 000 was the only value approved for use, will have filter value equal to 000, but following the ratification of this standard, the filter value should be set to match the object to which the tag is affixed, and use 000 only if the filter value for such object does not exist in the specification.

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

951

Table 9. SSCC Filter Values

952

- The *Partition* is an indication of where the subsequent Company Prefix and Serial Reference numbers are divided. This organization matches the structure in the

953

954
955
956
957
958
959

EAN.UCC SSCC in which the Company Prefix added to the Serial Reference number (prefixed by the single Extension Digit) totals 17 digits, yet the Company Prefix may vary from 6 to 12 digits and the Serial Reference from 11 to 5 digits. Table 10 shows allowed values of the partition value and the corresponding lengths of the company prefix and serial reference.

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

Table 10. SSCC-96 Partitions.

960
961
962
963
964
965
966
967
968
969
970
971
972
973

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

2.6.1.1 SSCC-96 Encoding Procedure

The following procedure creates an SSCC-96 encoding.
Given:

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

978 • The length L of the Company Prefix portion of the SSCC

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

980 Procedure:

981 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
982 the Partition Table (Table 10) to determine the Partition Value, P , the number of bits M in
983 the Company Prefix field, and the number of bits N in the Extension Digit and the Serial
984 Reference. If L is not found in any row of Table 10, stop: this SSCC cannot be encoded in
985 an SSCC-96.

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

988 3. Construct the Extension Digit and the Serial Reference by concatenating digits
989 $d_1d_{(L+2)}d_{(L+3)}\dots d_{17}$ and considering the result to be a decimal integer, S .

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

994 **2.6.1.2 SSCC-96 Decoding Procedure**

995 Given:

996 • An SSCC-96 as a 96-bit bit string $00110001b_{87}b_{86}\dots b_0$ (where the first eight bits
997 00110001 are the header)

998 Yields:

999 • An EAN.UCC SSCC

1000 • A Filter Value

1001 Procedure:

1002 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.

1003 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
1004 $P = 7$, stop: this bit string cannot be decoded as an SSCC-96.

1005 3. Look up the Partition Value P in Table 10 to obtain the number of bits M in the Company
1006 Prefix and the number of digits L in the Company Prefix.

1007 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned integer.
1008 If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal SSCC-96
1009 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$, adding leading
1010 zeros as necessary to make up L digits in total.

1011 5. Extract the Serial Reference by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{24}$ as an unsigned integer.
1012 If this integer is greater than or equal to $10^{(17-L)}$, stop: the input bit string is not a legal
1013 SSCC-96 encoding. Otherwise, convert this integer to a $(17-L)$ -digit decimal number
1014 $i_1i_2\dots i_{(17-L)}$, adding leading zeros as necessary to make $(17-L)$ digits.

- 1015 6. Construct a 17-digit number $d_1d_2\dots d_{17}$ where $d_1 = s_1$ from Step 5, $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$
 1016 from Step 4, and $d_{(L+2)}d_{(L+3)}\dots d_{17} = i_2i_3\dots i_{(17-L)}$ from Step 5.
- 1017 7. Calculate the check digit $d_{18} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13} + d_{15} + d_{17}) - (d_2 + d_4$
 1018 $+ d_6 + d_8 + d_{10} + d_{12} + d_{14} + d_{16})) \bmod 10$.
- 1019 8. The EAN.UCC SSCC is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{18}$.

1020 **2.7 Serialized Global Location Number (SGLN)**

1021 The EPC Tag Encoding scheme for GLN permits the direct embedding of the EAN.UCC
 1022 System standard GLN on EPC tags. EAN.UCC has defined the GLN as AI (414) and has
 1023 defined a GLN Extension Component as AI (254). The AI (254) uses the Set of Characters
 1024 defined in Appendix G.

1025 The use of the GLN Extension Component is intended for internal company purposes. For
 1026 communication between trading partners a GLN will be used. Trading partners can only use
 1027 the GLN Extension through mutual agreement but would have to establish an “out of band”
 1028 exchange of master data describing the extensions. If the GLN only encoding is used, then
 1029 the *Extension Component* shall be set to a fixed value of binary “0” for SGLN-96 and to
 1030 binary 0110000 followed by 133 binary “0” bits for SGLN-195 encoding as described in the
 1031 following SGLN procedures. In all cases the check digit is not encoded.

1032 **2.7.1 SGLN-96**

1033 In addition to a Header, the SGLN-96 is composed of five fields: the *Filter Value*, *Partition*,
 1034 *Company Prefix*, *Location Reference*, and *Extension Component*, as shown in Table 11.

	Header	Filter Value	Partition	Company Prefix	Location Reference	Extension Component
SGLN-96	8	3	3	20-40	21-1	41
	0011 0010 (Binary value)	(Refer to Table 12 for values)	(Refer to Table 13 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	999,999 – 0 (Max. decimal range*)	999,999,999,999(Max Decimal Value allowed) Minimum Decimal value=1 Reserved=0 All bits shall be set to 0 when an Extension Component is not encoded signifying GLN only.

1035
1036

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

1037
1038
1039
1040
1041

Table 11. The EPC SGLN-96 bit allocation, header, and maximum decimal values.

- *Header* is 8-bits, with a binary value of 0011 0010.
- *Filter Value* is not part of the GLN or EPC identifier, but is used for fast filtering and pre-selection of basic location types. The Filter Values for an SGLN is shown in Table 12 below.

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

1042
1043

Table 12. SGLN Filter Values .

1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064

- *Partition* is an indication of where the subsequent Company Prefix and Location Reference numbers are divided. This organization matches the structure in the EAN.UCC GLN in which the Company Prefix added to the Location Reference number totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the Location Reference number from 6 to 0 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Location Reference* fields are defined in Table 13.
- *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.
- *Location Reference*, if present, encodes the GLN Location Reference number.
- *Extension Component* contains a serial number. If an *Extension Component* is not used this value shall be set to a binary value of 0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000. The SGLN-96 encoding is only capable of representing integer-valued Extension Components with limited range. The EAN.UCC specifications permit a broader range of Extension Components. The EAN.UCC-128 barcode symbology provides for a 20-character alphanumeric Extension Component to be associated with a GLN using Application Identifier (AI) 254 [EAN.UCCGS]. It is possible to convert between the Extension Component in the SGLN-96 tag encoding and the Extension Component in AI 254 barcodes under certain conditions. Specifically, such interconversion is possible when the alphanumeric Extension Component in AI 254 happens to consist only of digits, with no leading zeros, and whose value when interpreted as an integer falls within the range limitations of the

1065 SGLN-96 tag encoding. These considerations are reflected in the encoding and
 1066 decoding procedures below.

1067

Partition Value (<i>P</i>)	Company Prefix		Location Reference	
	Bits (<i>M</i>)	Digits (<i>L</i>)	Bits (<i>N</i>)	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

Table 13. SGLN Partitions.

1068

2.7.1.1 SGLN-96 Encoding Procedure

1069

The following procedure creates an SGLN-96 encoding.

1070

Given:

1071

- 1072 • An EAN.UCC GLN consisting of digits $d_1d_2\dots d_{13}$
- 1073 • The length L of the Company Prefix portion of the GLN
- 1074 • An Extension Component S where $0 \leq S < 2^{40}$, or an EAN.UCC-128 Application
 1075 Identifier 254 consisting of characters $s_1s_2\dots s_K$, When the Extension Component S is 0,
 1076 the Encoding will be considered as a GLN only.

1077

- 1078 • A Filter Value F where $0 \leq F < 8$

1078

Procedure:

1079

- 1080 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
 1081 the Partition Table (Table 13) to determine the Partition Value, P , the number of bits M in
 1082 the Company Prefix field, and the number of bits N in the Location Reference field. If L is
 1083 not found in any row of Table 13, stop: this GLN cannot be encoded in an SGLN-96.
- 1084 2. Construct the Company Prefix by concatenating digits $d_1d_2\dots d_L$ and considering the result
 1085 to be a decimal integer, C .

1084

1085

- 1086 3. If $L < 12$ construct the Location Reference by concatenating digits $d_{(L+1)}d_{(L+2)}\dots d_{12}$ and
 1087 considering the result to be a decimal integer, I . If $L = 12$ set b_{41} to 0 since there is no
 1088 Location Reference digit.
- 1089 4. When the Extension Component is provided directly as an integer S where $0 \leq S < 2^{40}$,
 1090 proceed to Step 5. Otherwise, when the Extension Component is provided as an EAN.UCC-
 1091 128 Application Identifier 254 consisting of characters $s_1s_2\dots s_K$, construct the Extension
 1092 Component by concatenating characters $s_1s_2\dots s_K$. If any of these characters is not a digit,
 1093 stop: this Extension Component cannot be encoded in the SGLN-96 encoding. Also, if $K >$
 1094 1 and $s_1 = 0$, stop: this Extension Component cannot be encoded in the SGLN-96 encoding
 1095 (because leading zeros are not permitted except in the case where the Extension Component
 1096 consists of a single zero digit). Otherwise, consider the result to be a decimal integer, S . If S
 1097 $\geq 2^{40}$, stop: this Extension Component cannot be encoded in the SGLN-96 encoding.
- 1098 5. Construct the final encoding by concatenating the following bit fields, from most
 1099 significant to least significant: Header 00110010 (8 bits), Filter Value F (3 bits), Partition
 1100 Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Location Reference I
 1101 from Step 3 (N bits), Extension Component S from Step 4 (41 bits). Note that $M+N = 41$ bits
 1102 for all P .

1103 2.7.1.2 SGLN-96 Decoding Procedure

1104 Given:

- 1105 • An SGLN-96 as a 96-bit bit string $00110010b_{87}b_{86}\dots b_0$ (where the first eight bits
 1106 00110010 are the header)

1107 Yields:

- 1108 • An EAN.UCC GLN
- 1109 • An Extension Component
- 1110 • A Filter Value

1111 Procedure:

- 1112 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.
- 1113 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
 1114 $P = 7$, stop: this bit string cannot be decoded as an SGLN-96.
- 1115 3. Look up the Partition Value P in Table 13 to obtain the number of bits M in the Company
 1116 Prefix and the number of digits L in the Company Prefix.
- 1117 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned integer.
 1118 If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal SGLN-96
 1119 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$, adding leading
 1120 zeros as necessary to make up L digits in total.
- 1121 5. If $L < 12$ extract the Location Reference by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{41}$ as an
 1122 unsigned integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string
 1123 is not a legal SGLN-96 encoding. Otherwise, convert this integer to a $(12-L)$ -digit decimal
 1124 number $i_1i_2\dots i_{(12-L)}$, adding leading zeros as necessary to make $(12-L)$ digits.

- 1125 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 if $L <$
 1126 12 $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_1 i_2\dots i_{(12-L)}$ from Step 5.
- 1127 7. Calculate the check digit $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$
 1128 $d_{11})) \bmod 10$.
- 1129 8. The EAN.UCC GLN is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{13}$.
- 1130 9. Bits $b_{40}b_{39}\dots b_0$, considered as an unsigned integer, are the *Extension Component*.
- 1131 10. (Optional) If it is desired to represent the Extension Component as a EAN.UCC-128
 1132 Application Identifier 254, convert the integer from Step 9 to a decimal string with no
 1133 leading zeros. If the integer in Step 9 is zero, convert it to a string consisting of the single
 1134 character “0”.

1135 **2.7.2 SGLN-195**

1136 In addition to a Header, the SGLN-195 is composed of five fields: the *Filter Value*, *Partition*,
 1137 *Company Prefix*, *Location Reference*, and *Extension Component*, as shown in Table 14.

	Header	Filter Value	Partition	Company Prefix	Location Reference	Extension Component
SGLN-195	8	3	3	20-40	21-1	140
	0011 1001 (Binary value)	(Refer to Table 12 for values)	(Refer to Table 13 for values)	999,999 – 9,999 (Max. decimal range*)	999,999 – 0 (Max. decimal range*)	Up to 20 alphanumeric characters If the Extension Component is not used this value must be set to 0110000 followed by 133 binary 0 bits.

1138 *Max. decimal value range of Company Prefix and Location Reference fields vary according to contents of the
 1139 Partition field.

1140 **Table 14.** The EPC SGLN-195 bit allocation, header, and maximum decimal values.

- 1141 • *Header* is 8-bits, with a binary value of 0011 1001.
- 1142 • *Filter Value* is not part of the GLN or EPC identifier, but is used for fast filtering and
 1143 pre-selection of basic location types. The Filter Values for an SGLN is shown in Table
 1144 12.
- 1145 • *Partition* is an indication of where the subsequent Company Prefix and Location
 1146 Reference numbers are divided. This organization matches the structure in the
 1147 EAN.UCC GLN in which the Company Prefix added to the Location Reference
 1148 number totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the
 1149 Location Reference number from 6 to 0 digit(s). The available values of *Partition* and
 1150 the corresponding sizes of the *Company Prefix* and *Location Reference* fields are
 1151 defined in Table 13.
- 1152 • *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.

- 1153 • *Location Reference*, if present, encodes the GLN Location Reference number.
- 1154 • *ExtensionComponent* contains a serial number. If an *Extension Component* is not used
1155 signifying a GLN only, then this value shall be set to binary 0110000 followed by 133
1156 binary “0” bits. SGLN.-195 encoding is capable of representing alphanumeric
1157 Extension Component of up to 20 characters, permitting the full range of Extension
1158 Component available in the EAN.UCC-128 barcode symbology using Application
1159 Identifier (AI) 254 [EAN.UCCGS]. See Appendix G for permitted values.

1160 2.7.2.1 SGLN-195 Encoding Procedure

1161 The following procedure creates an SGLN-195 encoding.

1162 Given:

- 1163 • An EAN.UCC GLN consisting of digits $d_1d_2\dots d_{13}$
- 1164 • The length L of the Company Prefix portion of the GLN
- 1165 • An EAN.UCC-128 Application Identifier 254 consisting of characters $s_1s_2\dots s_K$, where K
1166 ≤ 20 .,. If the Application Identifier 254 consists of a single character 0 where $K=1$, this
1167 Encoding is considered to be a GLN only.
- 1168 • A Filter Value F where $0 \leq F < 8$

1169 Procedure:

- 1170 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
1171 the Partition Table (Table 13) to determine the Partition Value, P , the number of bits M in
1172 the Company Prefix field, and the number of bits N in the Location Reference field. If L is
1173 not found in any row of Table 13, stop: this GLN cannot be encoded in an SGLN-195.
- 1174 2. Construct the Company Prefix by concatenating digits $d_1d_2\dots d_L$ and considering the result
1175 to be a decimal integer, C .
- 1176 3. If $L < 12$ construct the Location Reference by concatenating digits $d_{(L+1)}d_{(L+2)}\dots d_{12}$ and
1177 considering the result to be a decimal integer, I . If $L = 12$ set b_{140} to 0 since there is no
1178 Location Reference digit.
- 1179 4. . Check that each of the characters $s_1s_2\dots s_K$ is one of the 82 characters listed in the table
1180 in Appendix G. If this is not the case, stop: this character string cannot be encoded as an
1181 SGLN-195. Otherwise construct the Extension Component by concatenating the 7-bit code,
1182 as given in Appendix G, for each of the characters $s_1s_2\dots s_K$, yielding $7K$ bits total. If $K < 20$,
1183 concatenate additional zero bits to the right to make a total of 140 bits.
- 1184 5. Construct the final encoding by concatenating the following bit fields, from most
1185 significant to least significant: Header 00111001 (8 bits), Filter Value F (3 bits), Partition
1186 Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Location Reference I
1187 from Step 3 (N bits), Extension Component S from Step 4 (140 bits). Note that $M+N =$
1188 41 bits for all P .

1189 **2.7.2.2 SGLN-195 Decoding Procedure**

1190 Given:

- 1191 • An SGLN-195 as a 195-bit bit string $00111001b_{186}b_{185}\dots b_0$ (where the first eight bits
1192 00111001 are the header)

1193 Yields:

- 1194 • An EAN.UCC GLN
1195 • An Extension Component
1196 • A Filter Value

1197 Procedure:

- 1198 1. Bits $b_{186}b_{185}b_{184}$, considered as an unsigned integer, are the Filter Value.
1199 2. Extract the Partition Value P by considering bits $b_{183}b_{182}b_{181}$ as an unsigned integer. If
1200 $P = 7$, stop: this bit string cannot be decoded as an SGLN-195.
1201 3. Look up the Partition Value P in Table 13 to obtain the number of bits M in the Company
1202 Prefix and the number of digits L in the Company Prefix.
1203 4. Extract the Company Prefix C by considering bits $b_{180}b_{179}\dots b_{(181-M)}$ as an unsigned
1204 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal
1205 SGLN-195 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$,
1206 adding leading zeros as necessary to make up L digits in total.
1207 5. When $L < 12$ extract the Location Reference by considering bits $b_{(180-M)} b_{(179-M)}\dots b_{140}$ as
1208 an unsigned integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit
1209 string is not a legal SGLN-195 encoding. Otherwise, convert this integer to a $(12-L)$ -digit
1210 decimal number $i_1i_2\dots i_{(12-L)}$, adding leading zeros as necessary to make $(12-L)$ digits.
1211 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 if $L <$
1212 12 $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_2 i_3\dots i_{(12-L)}$ from Step 5.
1213 7. Calculate the check digit $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$
1214 $d_{11})) \bmod 10$.
1215 8. The EAN.UCC GLN is the concatenation of digits from Steps 6 and 7: $d_1d_2\dots d_{13}$.
1216 9. Divide the remaining bits $b_{139}b_{138}\dots b_0$ into 7-bit segments. The result should consist of K
1217 non-zero binary segments followed by $20-K$ binary zero segments. If this is not the case,
1218 stop: this bit string cannot be decoded as an SGLN-195. Otherwise, look up each of the
1219 non-zero 7-bit segments in Appendix G to obtain a corresponding character. If any of the
1220 non-zero 7-bit segments has a value that is not in Appendix G, stop: this bit string cannot be
1221 decoded as an SGLN-195. If $K=1$ and $s_1=0$, then this indicates a GLN only with no
1222 *Extension Component*. Otherwise, the K characters so obtained, considered as a character
1223 string $s_1s_2\dots s_K$, is the value of the EAN.UCC AI 254.
1224 10. The EAN.UCC SGLN-195 is the concatenation of the digits from Steps 6 and 7 and the
1225 characters from Step 9. $\therefore d_1d_2\dots d_{13} s_1s_2\dots s_K$

1226

1227 **2.8 Global Returnable Asset Identifier (GRAI)**

1228 The EPC Tag Encoding scheme for GRAI permits the direct embedding of EAN.UCC
 1229 System standard GRAI on EPC tags. In all cases, the check digit is not encoded.

1230 **2.8.1 GRAI-96**

1231 In addition to a Header, the GRAI-96 is composed of five fields: the *Filter Value*, *Partition*,
 1232 *Company Prefix*, *Asset Type*, and *Serial Number*, as shown in Table 15.

	Header	Filter Value	Partition	Company Prefix	Asset Type	Serial Number
GRAI-96	8	3	3	20-40	24-4	38
	0011 0011 (Binary value)	(Refer to Table 16 for values)	(Refer to Table 17 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	999,999 – 0 (Max. decimal range*)	274,877,906,943 (Max. decimal value)

1233 *Max. decimal value range of Company Prefix and Asset Type fields vary according to contents of the Partition
 1234 field.

1235 **Table 15.** The EPC GRAI-96 bit allocation, header, and maximum decimal values.

- 1236 • *Header* is 8-bits, with a binary value of 0011 0011.
- 1237 • *Filter Value* is not part of the GRAI or EPC identifier, but is used for fast filtering and
 1238 pre-selection of basic asset types. The Filter Values for 96-bit and 170-bit GRAI are
 1239 the same. See Table 16.

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

1240 **Table 16.** GRAI Filter Values

1241
1242
1243
1244
1245
1246

- *Partition* is an indication of where the subsequent Company Prefix and Asset Type numbers are divided. This organization matches the structure in the EAN.UCC GRAI in which the Company Prefix added to the Asset Type number totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the Asset Type from 6 to 0 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Asset Type* fields are defined in Table 17.

Partition Value (<i>P</i>)	Company Prefix		Asset Type	
	Bits (<i>M</i>)	Digits (<i>L</i>)	Bits (<i>N</i>)	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

1247
1248

Table 17. GRAI Partitions.

1249
1250
1251
1252
1253
1254
1255

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

1256

2.8.1.1 GRAI-96 Encoding Procedure

1257

The following procedure creates a GRAI-96 encoding.

1258

Given:

1259

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

1260

- The length *L* of the Company Prefix portion of the GRAI

1261

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

1262

Procedure:

- 1263 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
 1264 the Partition Table (Table 17) to determine the Partition Value, P , the number of bits M in
 1265 the Company Prefix field, and the number of bits N in Asset Type field. If L is not found in
 1266 any row of Table 17, stop: this GRAI cannot be encoded in a GRAI-96.
- 1267 2. Construct the Company Prefix by concatenating digits $d_2d_3\dots d_{(L+1)}$ and considering the
 1268 result to be a decimal integer, C .
- 1269 3. If $L < 12$ construct the Asset Type by concatenating digits $d_{(L+2)}d_{(L+3)}\dots d_{13}$ and
 1270 considering the result to be a decimal integer, I . Otherwise set bits $b_{41}, b_{40}, b_{39}, b_{38}$ to 0000.
- 1271 4. Construct the Serial Number by concatenating digits $d_{15}d_{16}\dots d_K$. If any of these
 1272 characters is not a digit, stop: this GRAI cannot be encoded in the GRAI-96 encoding.
 1273 Otherwise, consider the result to be a decimal integer, S . If $S \geq 2^{38}$, stop: this GRAI cannot
 1274 be encoded in the GRAI-96 encoding. Also, if $K > 15$ and $d_{15} = 0$, stop: this GRAI cannot be
 1275 encoded in the GRAI-96 encoding (because leading zeros are not permitted except in the
 1276 case where the Serial Number consists of a single zero digit).
- 1277 5. Construct the final encoding by concatenating the following bit fields, from most
 1278 significant to least significant: Header 00110011 (8 bits), Filter Value F (3 bits), Partition
 1279 Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Asset Type I from
 1280 Step 3 (N bits), Serial Number S from Step 4 (38 bits). Note that $M+N = 44$ bits for all P .

1281 2.8.1.2 GRAI-96 Decoding Procedure

1282 Given:

- 1283 • An GRAI-96 as a 96-bit bit string 00110011 $b_{87}b_{86}\dots b_0$ (where the first eight bits
 1284 00110011 are the header)

1285 Yields:

- 1286 • An EAN.UCC GRAI
- 1287 • A Filter Value

1288 Procedure:

- 1289 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.
- 1290 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
 1291 $P = 7$, stop: this bit string cannot be decoded as a GRAI-96.
- 1292 3. Look up the Partition Value P in Table 17 to obtain the number of bits M in the Company
 1293 Prefix and the number of digits L in the Company Prefix.
- 1294 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned integer.
 1295 If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal GRAI-96
 1296 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$, adding leading
 1297 zeros as necessary to make up L digits in total.
- 1298 5. If $L < 12$ extract the Asset Type by considering bits $b_{(81-M)}b_{(80-M)}\dots b_{38}$ as an unsigned
 1299 integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a

- 1300 legal GRAI-96 encoding. Otherwise, convert this integer to a (12-L)-digit decimal number
 1301 $i_1i_2\dots i_{(12-L)}$, adding leading zeros as necessary to make (12-L) digits.
- 1302 6. Construct a 13-digit number $0d_2d_3\dots d_{13}$ where $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$ from Step 4, and
 1303 $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_1 i_2\dots i_{(12-L)}$ from Step 5.
- 1304 7. Calculate the check digit $d_{14} = (-3(d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8 + d_{10}$
 1305 $+ d_{12})) \bmod 10$.
- 1306 8. Extract the Serial Number by considering bits $b_{37}b_{36}\dots b_0$ as an unsigned integer. Convert
 1307 this integer to a decimal number $d_{15}d_{16}\dots d_K$, with no leading zeros (exception: if the integer
 1308 is equal to zero, convert it to a single zero digit).
- 1309 9. The EAN.UCC GRAI is the concatenation of a single zero digit and the digits from Steps
 1310 6, 7, and 8: $0d_2d_3\dots d_K$.

1311 **2.8.2 GRAI-170**

1312 In addition to a Header, the GRAI-170 is composed of five fields: the *Filter Value*, *Partition*,
 1313 *Company Prefix*, *Asset Type*, and *Serial Number*, as shown in Table 18.

	Header	Filter Value	Partition	Company Prefix	Asset Type	Serial Number
GRAI-170	8	3	3	20-40	24-4	112
	0011 0111 (Binary value)	(Refer to Table 16 for values)	(Refer to Table 17 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	999,999 – 0 (Max. decimal range*)	Up to 16 alphanumeric characters

1314 *Max. decimal value range of Company Prefix and Asset Type fields vary according to contents of the Partition
 1315 field.

1316 **Table 18.** The EPC GRAI-170 bit allocation, header, and maximum decimal values.

- 1317 • *Header* is 8-bits, with a binary value of 0011 0111
- 1318 • *Filter Value* is not part of the GRAI or EPC identifier, but is used for fast filtering and
 1319 pre-selection of basic asset types. The Filter Values for 96-bit and 170-bit GRAI are
 1320 the same. See Table 16. This specification anticipates that valuable Filter Values will
 1321 be determined once there has been time to consider the possible use cases.
- 1322 • *Partition* is an indication of where the subsequent Company Prefix and Asset Type
 1323 numbers are divided. This organization matches the structure in the EAN.UCC GRAI
 1324 in which the Company Prefix added to the Asset Type number totals 12 digits, yet the
 1325 Company Prefix may vary from 6 to 12 digits and the Asset Type from 6 to 0 digit(s).

1326 The available values of *Partition* and the corresponding sizes of the *Company Prefix*
 1327 and *Asset Type* fields for 96-bit and 170-bit GRAI are defined in Table 17.

- 1328 • *Company Prefix* contains a literal embedding of the EAN.UCC Company Prefix.
- 1329 • *Asset Type, if present*, encodes the GRAI Asset Type number.
- 1330 • *Serial Number* contains a mandatory alphanumeric serial number. The GRAI-170
 1331 encoding is capable of representing alphanumeric serial numbers of up to 16 characters,
 1332 permitting the full range of serial numbers available in the EAN.UCC-128 barcode
 1333 symbology using Application Identifier (AI) 8003 [EAN.UCCGS].

1334 2.8.2.1 GRAI-170 Encoding Procedure

1335 The following procedure creates a GRAI-170 encoding.

1336 Given:

- 1337 • An EAN.UCC GRAI consisting of digits $0d_2d_3\dots d_{14}$, and a variable length alphanumeric
 1338 serial number $s_{15}s_{16}\dots s_K$ where $15 \leq K \leq 30$.
- 1339 • The length L of the Company Prefix portion of the GRAI
- 1340 • A Filter Value F where $0 \leq F < 8$

1341 Procedure:

- 1342 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
 1343 the Partition Table (Table 17) to determine the Partition Value, P , the number of bits M in
 1344 the Company Prefix field, and the number of bits N in Asset Type field. If L is not found in
 1345 any row of Table 17, stop: this GRAI cannot be encoded in a GRAI-96.
- 1346 2. Construct the Company Prefix by concatenating digits $d_2d_3\dots d_{(L+1)}$ and considering the
 1347 result to be a decimal integer, C .
- 1348 3. If $L < 12$ construct the Asset Type by concatenating digits $d_{(L+2)}d_{(L+3)}\dots d_{13}$ and
 1349 considering the result to be a decimal integer, I . Otherwise set bits $b_{115}, b_{114}, b_{113}, b_{112}$ to 0000.
- 1350 4. Check that each of the characters $s_{15}s_{16}\dots s_K$ is one of the 82 characters listed in the table
 1351 in Appendix G. If this is not the case, stop: this character string cannot be encoded as an
 1352 GRAI-170. Otherwise construct the Serial Number by concatenating the 7-bit code, as given
 1353 in Appendix G, for each of the characters $s_{15}s_{16}\dots s_K$, yielding $7*(K-14)$ bits total. If $K < 30$,
 1354 concatenate additional zero bits to the right to make a total of 112 bits.
- 1355 5. Construct the final encoding by concatenating the following bit fields, from most
 1356 significant to least significant: Header 00110111 (8 bits), Filter Value F (3 bits), Partition
 1357 Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Asset Type I from
 1358 Step 3 (N bits), Serial Number S from Step 4 (112 bits). Note that $M+N = 44$ bits for all P .

1359 2.8.2.2 GRAI-170 Decoding Procedure

1360 Given:

- 1361 • An GRAI-170 as a 170-bit bit string $00110111b_{161}b_{160}\dots b_0$ (where the first eight bits
 1362 00110111 are the header)
- 1363 Yields:
- 1364 • An EAN.UCC GRAI
- 1365 • A Filter Value
- 1366 Procedure:
- 1367 1. Bits $b_{161}b_{160}b_{159}$, considered as an unsigned integer, are the Filter Value.
- 1368 2. Extract the Partition Value P by considering bits $b_{158}b_{157}b_{156}$ as an unsigned integer. If
 1369 $P = 7$, stop: this bit string cannot be decoded as a GRAI-170.
- 1370 3. Look up the Partition Value P in Table 17 to obtain the number of bits M in the Company
 1371 Prefix and the number of digits L in the Company Prefix.
- 1372 4. Extract the Company Prefix C by considering bits $b_{155}b_{154}\dots b_{(156-M)}$ as an unsigned
 1373 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal
 1374 GRAI-170 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$,
 1375 adding leading zeros as necessary to make up L digits in total.
- 1376 5. If $L < 12$ extract the Asset Type by considering bits $b_{(155-M)}b_{(154-M)}\dots b_{112}$ as an unsigned
 1377 integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a
 1378 legal GRAI-170 encoding. Otherwise, convert this integer to a $(12-L)$ -digit decimal number
 1379 $i_1i_2\dots i_{(12-L)}$, adding leading zeros as necessary to make $(12-L)$ digits.
- 1380 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, and if
 1381 $L < 12$ $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_1i_2\dots i_{(12-L)}$ from Step 5.
- 1382 7. Calculate the check digit $d_{14} = (-3(d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8 + d_{10}$
 1383 $+ d_{12})) \bmod 10$.
- 1384 8. Divide the remaining bits $b_{111}b_{110}\dots b_0$ into 7-bit segments. This string should consist of
 1385 K non-zero segments followed by $16-K$ zero segments. If this is not the case, stop: this bit
 1386 string cannot be decoded as an GRAI-170. Otherwise, look up each of the non-zero 7-bit
 1387 segments in Appendix G to obtain a corresponding character. If any of the non-zero 7-bit
 1388 segments has a value that is not in Appendix G, stop: this bit string cannot be decoded as an
 1389 GRAI-170. Otherwise, the first K characters considered as a character string is the serial
 1390 number $s_{15}s_{16}\dots s_K$.
- 1391 9. The EAN.UCC GRAI is the concatenation of a single zero digit, the digits from Steps 6
 1392 and 7 and the characters from Step 8. : $0d_2d_3\dots d_{14} s_{15}s_{16}\dots s_K$
- 1393

1394 2.9 Global Individual Asset Identifier (GIAI)

1395 The EPC Tag Encoding scheme for GIAI permits the direct embedding of EAN.UCC System
 1396 standard GIAI codes on EPC tags.

1397 **2.9.1 GIAI-96**

1398 In addition to a Header, the EPC GIAI-96 is composed of four fields: the *Filter Value*,
 1399 *Partition*, *Company Prefix*, and *Individual Asset Reference*, as shown in Table 19.

1400

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

1401

1402
1403

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

1404 **Table 19.** The EPC 96-bit GIAI bit allocation, header, and maximum decimal values.

- 1405 • *Header* is 8-bits, with a binary value of 0011 0100.
- 1406 • *Filter Value* is not part of the GIAI or EPC identifier, but is used for fast filtering and
 1407 pre-selection of basic asset types. The Filter Values for 96-bit and 202-bit GIAI are the
 1408 same. See Table 20.

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

1409

Table 20. GIAI Filter Values

- 1410 • The *Partition* is an indication of where the subsequent Company Prefix and Individual
 1411 Asset Reference numbers are divided. This organization matches the structure in the
 1412 EAN.UCC GIAI in which the Company Prefix may vary from 6 to 12 digits. The
 1413 available values of *Partition* and the corresponding sizes of the *Company Prefix* and
 1414 *Asset Reference* fields are defined in Table 21.

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

1415 **Table 21.** GIAI-96 Partitions.

- 1416 • *Company Prefix* contains a literal embedding of the Company Prefix.
- 1417 • *Individual Asset Reference* is a mandatory unique number for each instance. The EPC
 1418 representation is only capable of representing a subset of asset references allowed in
 1419 the General EAN.UCC Specifications. The capacity of this asset reference is less than
 1420 the maximum EAN.UCC System specification for asset references, no leading zeros
 1421 are permitted, and only numbers are permitted.

1422 **2.9.1.1 GIAI-96 Encoding Procedure**

1423 The following procedure creates a GIAI-96 encoding.

1424 Given:

- 1425 • An EAN.UCC GIAI consisting of digits $d_1d_2\dots d_K$, where $K \leq 30$.
- 1426 • The length L of the Company Prefix portion of the GIAI
- 1427 • A Filter Value F where $0 \leq F < 8$

1428 Procedure:

- 1429 1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of
 1430 the Partition Table (Table 21) to determine the Partition Value, P , the number of bits M in
 1431 the Company Prefix field, and the number of bits N in the Individual Asset Reference field.
 1432 If L is not found in any row of Table 21, stop: this GIAI cannot be encoded in a GIAI-96.

- 1433 2. Construct the Company Prefix by concatenating digits $d_1d_2\dots d_L$ and considering the result
 1434 to be a decimal integer, C .
- 1435 3. Construct the Individual Asset Reference by concatenating digits $d_{(L+1)}d_{(L+2)}\dots d_K$. If any
 1436 of these characters is not a digit, stop: this GIAI cannot be encoded in the GIAI-96 encoding.
 1437 Otherwise, consider the result to be a decimal integer, S . If $S \geq 2^N$, stop: this GIAI cannot be
 1438 encoded in the GIAI-96 encoding. Also, if $K > L+1$ and $d_{(L+1)} = 0$, stop: this GIAI cannot be
 1439 encoded in the GIAI-96 encoding (because leading zeros are not permitted except in the case
 1440 where the Individual Asset Reference consists of a single zero digit).
- 1441 4. Construct the final encoding by concatenating the following bit fields, from most
 1442 significant to least significant: Header 00110100 (8 bits), Filter Value F (3 bits), Partition
 1443 Value P from Step 2 (3 bits), Company Prefix C from Step 3 (M bits), Individual Asset
 1444 Number S from Step 4 (N bits). Note that $M+N = 82$ bits for all P .

1445 2.9.1.2 GIAI-96 Decoding Procedure

1446 Given:

- 1447 • A GIAI-96 as a 96-bit bit string $00110100b_{87}b_{86}\dots b_0$ (where the first eight bits
 1448 00110100 are the header)

1449 Yields:

- 1450 • An EAN.UCC GIAI
- 1451 • A Filter Value

1452 Procedure:

- 1453 1. Bits $b_{87}b_{86}b_{85}$, considered as an unsigned integer, are the Filter Value.
- 1454 2. Extract the Partition Value P by considering bits $b_{84}b_{83}b_{82}$ as an unsigned integer. If
 1455 $P = 7$, stop: this bit string cannot be decoded as a GIAI-96.
- 1456 3. Look up the Partition Value P in Table 21 to obtain the number of bits M in the Company
 1457 Prefix and the number of digits L in the Company Prefix.
- 1458 4. Extract the Company Prefix C by considering bits $b_{81}b_{80}\dots b_{(82-M)}$ as an unsigned integer.
 1459 If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal GIAI-96
 1460 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$, adding leading
 1461 zeros as necessary to make up L digits in total.
- 1462 5. Extract the Individual Asset Reference by considering bits $b_{(81-M)}b_{(80-M)}\dots b_0$ as an
 1463 unsigned integer. If this integer is greater than or equal to $10^{(30-L)}$, stop: the input bit string
 1464 is not a legal GIAI-96 encoding. Otherwise, convert this integer to a decimal number
 1465 $s_1s_2\dots s_J$, with no leading zeros (exception: if the integer is equal to zero, convert it to a single
 1466 zero digit).
- 1467 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
 1468 $d_{(L+1)}d_{(L+2)}\dots d_K = s_1s_2\dots s_J$ from Step 5. This K -digit number, where $K \leq 30$, is the
 1469 EAN.UCC GIAI.

1470 **2.9.2 GIAI-202**

1471 In addition to a Header, the EPC GIAI-202 is composed of four fields: the *Filter Value*,
 1472 *Partition*, *Company Prefix*, and *Individual Asset Reference*, as shown in Table 22.

1473

	Header	Filter Value	Partition	Company Prefix	Individual Asset Reference
GIAI-202	8	3	3	20-40	168-126
	0011 1000 (Binary value)	(Refer to Table 20 for values)	(Refer to Table 21 for values)	999,999 – 999,999,999,999 (Max. decimal range*)	Up to 24 alphanumeric characters

1474

1475
1476

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

1477 **Table 22.** The EPC 202-bit GIAI bit allocation, header, and maximum decimal values.

- 1478 • *Header* is 8-bits, with a binary value of 0011 1000.
- 1479 • *Filter Value* is not part of the GIAI or EPC identifier, but is used for fast filtering and
 1480 pre-selection of basic asset types. The Filter Values for 96-bit and 202-bit GIAI are the
 1481 same. See Table 20.
- 1482 • The *Partition* is an indication of the size of the subsequent Company Prefix. This
 1483 organization matches the structure in the EAN.UCC GIAI in which the Company
 1484 Prefix may vary from 6 to 12 digits. The available values of *Partition* and the
 1485 corresponding size of the *Company Prefix* field is defined in Table 23.

1486

Partition Value (P)	Company Prefix		Individual Asset Reference	
	Bits (M)	Digits (L)	Bits (N)	Characters
0	40	12	148	18
1	37	11	151	19
2	34	10	154	20

Partition Value (<i>P</i>)	Company Prefix		Individual Asset Reference	
	Bits (<i>M</i>)	Digits (<i>L</i>)	Bits (<i>N</i>)	Characters
3	30	9	158	21
4	27	8	161	22
5	24	7	164	23
6	20	6	168	24

1487

1488

Table 23. GIAI-202 Partitions.

1489

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

1490

- *Individual Asset Reference* contains a mandatory alphanumeric asset reference number.

1491

The GIAI-202 encoding is capable of representing alphanumeric serial numbers of up

1492

to 24 characters, permitting the full range of serial numbers available in the EAN.UCC-

1493

128 barcode symbology using Application Identifier (AI) 8004 [EAN.UCCGS].

1494

- *Company Prefix* and *Individual Asset Reference* should never total more than 30

1495

characters.

1496

2.9.2.1 GIAI-202 Encoding Procedure

1497

1498

The following procedure creates a GIAI-202 encoding.

1499

Given:

1500

- An EAN.UCC GIAI consisting of digits $d_1d_2d_3\dots d_L$, and a variable length alphanumeric

1501

serial number $s_{L+1}s_{L+2}\dots s_K$ where $L+1 \leq K \leq 30$.

1502

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

1503

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

1504

Procedure:

1505

1. Look up the length L of the Company Prefix in the “Company Prefix Digits” column of

1506

the Partition Table (Table 23) to determine the Partition Value, P , the number of bits M in

1507

the Company Prefix field, and the number of bits N in the Individual Asset Reference field.

1508

If L is not found in any row of Table 23, stop: this GIAI cannot be encoded in a GIAI-202.

1509

2. Construct the Company Prefix by concatenating digits $d_1d_2\dots d_L$ and considering the result

1510

to be a decimal integer, C .

1511

3. Check that each of the characters $s_{(L+1)}s_{(L+2)}\dots s_K$ is one of the 82 characters listed in the

1512

table in Appendix G. If this is not the case, stop: this character string cannot be encoded as

1513 an GIAI-202. Otherwise construct the Individual Asset Reference by concatenating the 7-bit
1514 code, as given in Appendix G, for each of the characters $s_{(L+1)}s_{(L+2)}\dots s_K$ yielding $7*(K-L)$
1515 bits total. Concatenate additional zero bits to the right, if necessary, to make a total of $(188-$
1516 $M)$ bits, where M is the number of bits in the Company Prefix portion as determined in Step
1517 1.

1518 4. Construct the final encoding by concatenating the following bit fields, from most
1519 significant to least significant: Header 00111000 (8 bits), Filter Value F (3 bits), Partition
1520 Value P from Step 1 (3 bits), Company Prefix C from Step 2 (M bits), Individual Asset
1521 Number S from Step 3 ($188-M$ bits),

1522

1523 **2.9.2.2 GIAI-202 Decoding Procedure**

1524 Given:

- 1525 • A GIAI-202 as a 202-bit bit string $00111000b_{193}b_{192}\dots b_0$ (where the first eight bits
1526 00111000 are the header)

1527 Yields:

- 1528 • An EAN.UCC GIAI
- 1529 • A Filter Value

1530 Procedure:

- 1531 1. Bits $b_{193}b_{192}b_{191}$, considered as an unsigned integer, are the Filter Value.
- 1532 2. Extract the Partition Value P by considering bits $b_{190}b_{189}b_{188}$ as an unsigned integer. If
1533 $P = 7$, stop: this bit string cannot be decoded as a GIAI-202.
- 1534 3. Look up the Partition Value P in Table 23 to obtain the number of bits M in the Company
1535 Prefix and the number of digits L in the Company Prefix.
- 1536 4. Extract the Company Prefix C by considering bits $b_{187}b_{186}\dots b_{(188-M)}$ as an unsigned
1537 integer. If this integer is greater than or equal to 10^L , stop: the input bit string is not a legal
1538 GIAI-202 encoding. Otherwise, convert this integer into a decimal number $p_1p_2\dots p_L$, adding
1539 leading zeros as necessary to make up L digits in total.
- 1540 5. Extract the Individual Asset Reference by dividing the remaining bits $b_{(187-M)}b_{(186-M)}\dots b_0$
1541 into 7 bit segments beginning with the segment $b_{(187-M)}b_{(186-M)}\dots b_{(181-M)}$, and continuing as
1542 far as possible (there may be up to four bits left over at the end).. The result should consist
1543 of J non-zero segments followed by zero or more zero-valued segments, with any remaining
1544 bits also being zeros. If this is not the case, stop: this bit string cannot be decoded as a GIAI
1545 -202. Otherwise, look up each of the non-zero 7-bit segments in Appendix G to obtain a
1546 corresponding character. If any of the non-zero 7-bit segments has a value that is not in
1547 Appendix G, stop: this bit string cannot be decoded as a GIAI-202. Otherwise, the first J
1548 characters considered as a character string is the Asset Reference Number $s_{(1)}s_{(2)}\dots s_J$.
- 1549 6. Construct a K -character string $s_1s_2\dots s_K$ where $s_1s_2\dots s_L = p_1p_2\dots p_L$ from Step 4, and where
1550 $s_{(L+1)}s_{(L+2)}\dots s_K = s_{(1)}s_{(2)}\dots s_J$ from Step 5. This K -character string, where $K \leq 30$, is the
1551 EAN.UCC GIAI.

1552

1553 2.10 DoD Tag Data Constructs

1554 2.10.1 DoD-96

1555 This tag data construct may be used to encode Class 1 tags for shipping goods to the United
1556 States Department of Defense by an entity who has already been assigned a CAGE
1557 (Commercial and Government Entity) code.

1558 At the time of this writing, the details of what information to encode into these fields is
1559 explained in a document titled "United States Department of Defense Supplier's Passive
1560 RFID Information Guide" that can be obtained at the United States Department of Defense's
1561 web site (<http://www.dodrfid.org/supplierguide.htm>).

1562 The current encoding structure of DoD-96 Tag Data Construct is shown in Table 24 below.

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

1563

Table 24. The DoD-96 bit allocation, header, and maximum decimal values

1564

1565 3 URI Representation

1566 This section defines standards for the encoding of the Electronic Product Code™ as a
1567 Uniform Resource Identifier (URI). The URI Encoding complements the EPC Tag
1568 Encodings defined for use within RFID tags and other low-level architectural components.
1569 URIs provide a means for application software to manipulate Electronic Product Codes in a
1570 way that is independent of any particular tag-level representation, decoupling application
1571 logic from the way in which a particular Electronic Product Code was obtained from a tag.

1572 *Explanation (non-normative): The pure identity URI for a given EPC is the same regardless*
1573 *of the encoding. For example, the following pure identity URI*
1574 *urn:epc:id:sgtin:0064141.112345.400 is the same regardless of whether it is encoded into a*
1575 *tag as an SGTIN-96 or an SGTIN-198. Other representations than the pure identity URI for*
1576 *use above the reader or middleware layer shall not be used, because they can lead to*
1577 *misinterpretations in the information system. Exclusively on the reader layer and below the*
1578 *encoding schemes including header, filter value and partition must be considered for*
1579 *filtering or writing processes.*

1580 This section defines four categories of URI. The first are URIs for pure identities,
1581 sometimes called “canonical forms.” These contain only the unique information that
1582 identifies a specific physical object, and are independent of tag encodings. The second
1583 category is URIs that represent specific tag encodings. These are used in software
1584 applications where the encoding scheme is relevant, as when commanding software to write
1585 a tag. The third category is URIs that represent patterns, or sets of EPCs. These are used
1586 when instructing software how to filter tag data. The last category is a URI representation
1587 for raw tag information, generally used only for error reporting purposes.

1588 All categories of URIs are represented as Uniform Resource Names (URNs) as defined by
1589 [RFC2141], where the URN Namespace is *epc*.

1590 This section complements Section 3, EPC Bit-level Encodings, which specifies the currently
1591 defined tag-level representations of the Electronic Product Code.

1592 **3.1 URI Forms for Pure Identities**

1593 (This section is non-normative; the formal specifications for the URI types are given in
1594 Sections 3.2.4 and 5.)

1595 URI forms are provided for pure identities, which contain just the EPC fields that serve to
1596 distinguish one object from another. These URIs take the form of Uniform Resource Names
1597 (URNs), with a different URN namespace allocated for each pure identity type.

1598 For the EPC General Identifier (Section 2.1.1), the pure identity URI representation is as
1599 follows:

1600 *urn:epc:id:gid:GeneralManagerNumber.ObjectClass.SerialNumber*

1601 In this representation, the three fields *GeneralManagerNumber*, *ObjectClass*, and
1602 *SerialNumber* correspond to the three components of an EPC General Identifier as
1603 described in Section 2.1.1. In the URI representation, each field is expressed as a decimal
1604 integer, with no leading zeros (except where a field’s value is equal to zero, in which case a
1605 single zero digit is used).

1606 There are also pure identity URI forms defined for identity types corresponding to certain
1607 types within the EAN.UCC System family of codes as defined in Section 2.1.2; namely, the
1608 Serialized Global Trade Item Number (SGTIN), the Serial Shipping Container Code (SSCC),
1609 the Serialized Global Location Number (SGLN), the Global Reusable Asset Identifier
1610 (GRAI), and the Global Individual Asset Identifier (GIAI). The URI representations
1611 corresponding to these identifiers are as follows:

1612 *urn:epc:id:sgtin:CompanyPrefix.ItemReference.SerialNumber*

1613 *urn:epc:id:sscc:CompanyPrefix.SerialReference*

1614 *urn:epc:id:sgln:CompanyPrefix.LocationReference.ExtensionComponent*

1615 *urn:epc:id:grai:CompanyPrefix.AssetType.SerialNumber*

1616 *urn:epc:id:giai:CompanyPrefix.IndividualAssetReference*

1617 In these representations, *CompanyPrefix* corresponds to an EAN.UCC company prefix
1618 assigned to a manufacturer by GS1. (A UCC company prefix is converted to an EAN.UCC

1619 company prefix by adding one leading zero at the beginning.) The number of digits in this
1620 field is significant, and leading zeros are included as necessary.

1621 The *ItemReference*, *SerialReference*, *LocationReference*, and
1622 *AssetType* fields correspond to the similar fields of the GTIN, SSCC, GLN, and GRAI,
1623 respectively. Like the *CompanyPrefix* field, the number of digits in these fields is
1624 significant, and leading zeros are included as necessary. The number of digits in these fields,
1625 when added to the number of digits in the *CompanyPrefix* field, always total the same
1626 number of digits according to the identity type: 13 digits total for SGTIN, 17 digits total for
1627 SSCC, 12 digits total for SGLN, and 12 characters total for the GRAI. (The
1628 *ItemReference* field of the SGTIN includes the GTIN Indicator (PI) digit, appended to
1629 the beginning of the item reference. The *SerialReference* field includes the SSCC
1630 Extension Digit (ED), followed by the serial reference. In no case are check digits included
1631 in URI representations.)

1632 The *SerialNumber* field of the SGTIN and GRAI, the *ExtensionComponent* of the
1633 SGLN, as well as the *IndividualAssetReference* field of the GIAI, may include
1634 digits, letters, and certain other characters. In order for an SGTIN, SGLN, GRAI, or GIAI to
1635 be encoded on a 96-bit tag, however, these fields must consist only of digits with no leading
1636 zeros. These restrictions are defined in the encoding procedures for these types, as well as in
1637 Appendix F.

1638 An SGTIN, SSCC, etc in this form is said to be in SGTIN-URI form, SSCC-URI form, etc
1639 form, respectively. Here are examples:

1640 urn:epc:id:sgtin:0652642.800031.400
1641 urn:epc:id:sscc:0652642.0123456789
1642 urn:epc:id:sgln:0652642.12345.40 (Use this form when Extension
1643 Component is used)
1644 urn:epc:id:sgln:0652642.12345.0 (Use this form when Extension
1645 Component is not used)
1646 urn:epc:id:grai:0652642.12345.1234
1647 urn:epc:id:giai:0652642.123456

1648 Referring to the first example, the corresponding GTIN-14 code is 80652642000311. This
1649 divides as follows: the first digit (8) is the PI digit, which appears as the first digit of the
1650 *ItemReference* field in the URI, the next seven digits (0652642) are the
1651 *CompanyPrefix*, the next five digits (00031) are the remainder of the *ItemReference*,
1652 and the last digit (1) is the check digit, which is not included in the URI.

1653 Referring to the second example, the corresponding SSCC is 006526421234567896 and the
1654 last digit (6) is the check digit, not included in the URI.

1655 Referring to the third and fourth examples, the corresponding GLN is 0652642123458,
1656 where the last digit (8) is the check digit, not included in the URI.

1657 Referring to the fifth example, the corresponding GRAI is 006526421234581234, where the
1658 digit (8) is the check digit, not included in the URI.

1659 Referring to the sixth example, the corresponding GIAI is 0652642123456. (GIAI codes do
1660 not include a check digit.)

1661 Note that all six URI forms have an explicit indication of the division between the company
1662 prefix and the remainder of the code. This is necessary so that the URI representation may
1663 be converted into tag encodings. In general, the URI representation may be converted to the
1664 corresponding EAN.UCC numeric form (by combining digits and calculating the check
1665 digit), but converting from the EAN.UCC numeric form to the corresponding URI
1666 representation requires independent knowledge of the length of the company prefix.

1667 For the DoD identifier as defined in Section 3.9, the pure identity URI representation is as
1668 follows:

1669 `urn:epc:id:usdod:CAGECodeOrDODAAC.serialNumber`

1670 where *CAGECodeOrDODAAC* is the five-character CAGE code or six-character DoDAAC,
1671 and *serialNumber* is the serial number represented as a decimal integer with no leading
1672 zeros (except that a serial number whose value is zero should be represented as a single zero
1673 digit). Note that a space character is never included as part of *CAGECodeOrDODAAC* in the
1674 URI form, even though on a 96-bit tag a space character is used to pad the five-character
1675 CAGE code to fit into the six-character field on the tag.

1676

1677 **3.2 URI Forms for Related Data Types**

1678 (This section is non-normative; the formal specifications for the URI types are given in
1679 Sections 4.3 and Section 5.)

1680 There are several data types that commonly occur in applications that manipulate Electronic
1681 Product Codes, which are not themselves Electronic Product Codes but are closely related.
1682 This specification provides URI forms for those as well. The general form of the *epc* URN
1683 Namespace is

1684 `urn:epc:type:typeSpecificPart`

1685 The *type* field identifies a particular data type, and *typeSpecificPart* encodes
1686 information appropriate for that data type. Currently, there are three possibilities defined for
1687 *type*, discussed in the next three sections.

1688 **3.2.1 URIs for EPC Tags**

1689 In some cases, it is desirable to encode in URI form a specific tag encoding of an EPC. For
1690 example, an application may wish to report to an operator what kinds of tags have been read.
1691 In another example, an application responsible for programming tags needs to be told not
1692 only what Electronic Product Code to put on a tag, but also the encoding scheme to be used.
1693 Finally, applications that wish to manipulate any additional data fields on tags need some
1694 representation other than the pure identity forms.

1695 EPC Tag URIs are encoded by setting the *type* field to *tag*, with the entire URI having
1696 this form:

1697 `urn:epc:tag:EncName:EncodingSpecificFields`

1698 where *EncName* is the name of an EPC Tag Encoding scheme, and
1699 *EncodingSpecificFields* denotes the data fields required by that encoding scheme,
1700 separated by dot characters. Exactly what fields are present depends on the specific
1701 encoding scheme used.

1702 In general, there are one or more encoding schemes (and corresponding *EncName* values)
1703 defined for each pure identity type. For example, the SGTIN Identifier has two encodings
1704 defined: `sgtin-96` and `sgtin-198`, corresponding to the 96-bit encoding and the 198-
1705 bit encoding. Note that these encoding scheme names are in one-to-one correspondence with
1706 unique tag Header values, which are used to represent the encoding schemes on the tag itself.

1707 The *EncodingSpecificFields*, in general, include all the fields of the corresponding
1708 pure identity type, possibly with additional restrictions on numeric range, plus additional
1709 fields supported by the encoding. For example, all of the defined encodings for the
1710 Serialized GTIN include an additional Filter Value that applications use to do tag filtering
1711 based on object characteristics associated with (but not encoded within) an object's pure
1712 identity.

1713 Here is an example: a Serialized GTIN 96-bit encoding:

1714 `urn:epc:tag:sgtin-96:3.0652642.800031.400`

1715 In this example, the number 3 is the Filter Value.

1716 The tag URI for the DoD identifier is as follows:

1717 `urn:epc:tag:tagType:filter.CAGECodeOrDODAAC.serialNumber`

1718 where *tagType* is `usdod-96`, *filter* is the filter value represented as two decimal
1719 digits, and the other two fields are as defined above in 4.1.

1720

1721 **3.2.2 URIs for Raw Bit Strings Arising From Invalid Tags**

1722 Certain bit strings do not correspond to legal encodings. Here are several examples:

- 1723 • If the most significant bits of a bit string cannot be recognized as a valid EPC header, the
1724 bit-level pattern is not a legal EPC Tag Encoding.
- 1725 • If the most significant bits of a bit string are recognized as a valid EPC header, but the
1726 binary value of a field in the corresponding tag encoding is greater than the value that
1727 can be contained in the number of decimal digits in that field in the URI form, the bit
1728 level pattern is not a legal EPC Tag Encoding.
- 1729 • A Gen 2 Tag whose “toggle bit” is set to one (see Section 3.2) by definition does not
1730 contain an EPC Tag Encoding.

1731 While in these situations a bit string is not a legal EPC Tag Encoding, software may wish to
1732 report such invalid bit-level patterns to users or to other software. For such cases, a
1733 representation of invalid bit-level patterns as URIs is provided. The *raw* form of the URI has
1734 this general form:

1735 urn:epc:raw:BitLength.Value

1736 where *BitLength* is the number of bits in the invalid representation, and *Value* is the
1737 entire bit-level representation converted to a single hexadecimal number and preceded by the
1738 letter “x”. For example, this bit string:

1739 0000000000000000000000000100100011010011011110101011011011111011101111

1740 which is invalid because no valid header begins with 0000 0000, corresponds to this raw
1741 URI:

1742 urn:epc:raw:64.x00001234DEADBEEF

1743 In order to ensure that a given bit string has only one possible raw URI representation, the
1744 number of digits in the hexadecimal value is required to be equal to the *BitLength* divided
1745 by four and rounded up to the nearest whole number. Moreover, only uppercase letters are
1746 permitted for the hexadecimal digits A, B, C, D, E, and F.

1747 It is intended that this URI form be used only when reporting errors associated with reading
1748 invalid tags and when representing partially written tag. It is *not* intended to be a general
1749 mechanism for communicating arbitrary bit strings for other purposes.

1750 *Explanation (non-normative): The reason for recommending against using the raw URI for*
1751 *general purposes is to avoid having an alternative representation for legal tag encodings.*

1752 Earlier versions of this specification described a decimal, as opposed to hexadecimal, version
1753 of the raw URI. This is still supported for back-compatibility, but its use is no longer
1754 recommended. The “x” character is included so that software may distinguish between the
1755 decimal and hexadecimal forms.

1756 3.2.2.1 Use of the Raw URI with Gen 2 Tags

1757 The EPC memory of a Gen 2 Tag may contain either an EPC Tag Encoding or a value from
1758 a different numbering system for which an ISO Application Family Identifier (AFI) has been
1759 assigned. The “toggle” bit (bit 17x) of EPC memory distinguishes between these two
1760 possibilities (see Section 2.2).

1761 The Raw URI as described above is intended primarily to represent undecodable EPC Tag
1762 Encodings or partially written tags. For a Gen 2 Tag, therefore, the Raw URI described
1763 above is used only when the toggle bit is a zero, indicating that the tag is supposed to contain
1764 an EPC Tag Encoding.

1765 For completeness, an alternative form of the Raw URI is provided to represent the contents
1766 of a UHF Class 1 Gen 2 Tag whose toggle bit is a one. It has the following form:

1767 urn:epc:raw:BitLength.AFI.Value

1768 where *BitLength* is the number of bits in the non-EPC representation (not including the
1769 AFI), AFI is the Application Family Identifier represented as a two-digit hexadecimal
1770 number and preceded by the letter “x”, and *Value* is the remainder of EPC memory
1771 converted to a single hexadecimal number and preceded by the letter “x”.

1772 **3.2.2.2 The Length Field of a Raw URI when using Gen 2 Tags (non-normative)**

1773 (This non-normative section explains a subtle interaction between the Raw URI and the
1774 length indication on Gen 2 Tags.)

1775 Unlike earlier generations of RFID tags, the Gen 2 Tag is designed so that the length of the
1776 EPC Tag Encoding stored on the tag is not necessarily the same as the total length of EPC
1777 memory provided. The Gen 2 Specification provides a five-bit length indication, that
1778 indicates the length of the EPC memory to the nearest multiple of 16 bits (see Section 2.2.2).

1779 Because of the way the EPC Tag Encoding aligns in the Gen 2 Tag's EPC memory, the five-
1780 bit length indication does not necessarily indicate the length of the EPC Tag Encoding. This
1781 is because the length indication is limited to expressing multiples of 16 bits, including the
1782 first 16 bits in the protocol control (PC) bits which is not part of the EPC Tag Encoding. For
1783 example, if a Gen 2 Tag contains an SGTIN-198 EPC, the EPC Tag Encoding is 198 bits,
1784 which means there are total of 214 bits is considered when calculating the length indicator
1785 (198 EPC Tag Encoding bits plus the 16 PC bits). The nearest round up length indicator
1786 value is 01101 (binary), which indicates a total length of 224 bits. Working in the other
1787 direction, if a length indicator of 01101 is read from a Gen 2 Tag, it indicates a total of 224
1788 bits including the 16 PC bits, and therefore appears to indicate an EPC Tag Encoding of 208
1789 bits.

1790 This does not present a problem when a Gen 2 Tag contains a valid EPC. The procedures in
1791 Sections 4.3 and 4.4 use the header table in Section 2.1 to determine the length of the EPC,
1792 and discard any extra bits that may be implied by the length indication. When the contents
1793 of a Gen 2 Tag are converted to a Raw URI, however, the length indication on the tag is used
1794 to calculate the length in the URI. Therefore the length representation in the raw URI will
1795 have different bit length to the EPC Tag Encoding bits. Also one must consider the fact that
1796 value field in the raw URI may be different, because the values from Gen 2 tags may also
1797 include excess bits that are filled with zeros up to the word boundary.

1798 For these and other reasons, Raw URIs should never be used within information systems to
1799 represent valid EPCs.

1800 **3.2.3 URIs for EPC Patterns**

1801 Certain software applications need to specify rules for filtering lists of tags according to
1802 various criteria. This specification provides a *pattern* URI form for this purpose. A pattern
1803 URI does not represent a single tag encoding, but rather refers to a set of tag encodings. A
1804 typical pattern looks like this:

1805 `urn:epc:pat:sgtin-96:3.0652642.[102400-204700].*`

1806 This pattern refers to any EPC SGTIN Identifier 96-bit tag, whose Filter field is 3, whose
1807 Company Prefix is 0652642, whose Item Reference is in the range $102400 \leq \textit{itemReference}$
1808 ≤ 204700 , and whose Serial Number may be anything at all.

1809 In general, there is a pattern form corresponding to each tag encoding form (Section 3.2.1),
1810 whose syntax is essentially identical except that ranges or the star (*) character may be used
1811 in each field.

1812 For the SGTIN, SSCC, SGLN, GRAI and GIAI patterns, the pattern syntax slightly restricts
1813 how wildcards and ranges may be combined. Only two possibilities are permitted for the
1814 *CompanyPrefix* field. One, it may be a star (*), in which case the following field
1815 (*ItemReference*, *SerialReference*, *LocationReference*, *AssetType* or
1816 *IndividualAssetReference*) must also be a star. Two, it may be a specific company
1817 prefix, in which case the following field may be a number, a range, or a star. A range may
1818 not be specified for the *CompanyPrefix*.

1819 *Explanation (non-normative): Because the company prefix is variable length, a range may*
1820 *not be specified, as the range might span different lengths. When a particular company*
1821 *prefix is specified, however, it is possible to match ranges or all values of the following field,*
1822 *because its length is fixed for a given company prefix. The other case that is allowed is when*
1823 *both fields are a star, which works for all tag encodings because the corresponding tag*
1824 *fields (including the Partition field, where present) are simply ignored.*

1825 The pattern URI for the DoD Construct is as follows:

1826 `urn:epc:pat:tagType:filterPat.CAGECodeOrDODAACPat.serialNumber`
1827 `Pat`

1828 where *tagType* is as defined above in 4.2.1, *filterPat* is either a filter value, a range of
1829 the form [*lo-hi*], or a * character; *CAGECodeOrDODAACPat* is either a CAGE
1830 Code/DODAAC or a * character; and *serialNumberPat* is either a serial number, a
1831 range of the form [*lo-hi*], or a * character.

1832 **3.2.4 URIs for EPC Pure Identity Patterns**

1833 Certain software applications need to specify rules for filtering lists of EPC pure identities
1834 according to various criteria. This specification provides a *pure identity pattern* URI form
1835 for this purpose. A pure identity pattern URI does not represent a single EPC, but rather
1836 refers to a set of EPCs. A typical pure identity pattern looks like this:

1837 `urn:epc:idpat:sgtin:0652642.*.*`

1838 This pattern refers to any EPC SGTIN, whose Company Prefix is 0652642, whose Item
1839 Reference and Serial Number may be anything at all. The tag length and filter bits are not
1840 considered at all in matching the pattern to EPCs.

1841 In general, there is a pattern form corresponding to each pure identity form (Section 3.1),
1842 whose syntax is essentially identical except any number of fields starting at the right may be
1843 a star (*). This is more restrictive than tag patterns (Section 3.2.3), in that the star characters
1844 must occupy adjacent rightmost fields and the range syntax is not allowed at all.

1845 The pure identity pattern URI for the DoD Construct is as follows:

1846 `urn:epc:idpat:usdod:CAGECodeOrDODAACPat.serialNumberPat`

1847 with similar restrictions on the use of star (*).

1848 **3.3 Syntax**

1849 The syntax of the EPC-URI and the URI forms for related data types are defined by the
1850 following grammar.

1851 **3.3.1 Common Grammar Elements**

1852 NumericComponent ::= ZeroComponent | NonZeroComponent
1853 ZeroComponent ::= "0"
1854 NonZeroComponent ::= NonZeroDigit Digit*
1855 PaddedNumericComponent ::= Digit+
1856 Digit ::= "0" | NonZeroDigit
1857 NonZeroDigit ::= "1" | "2" | "3" | "4"
1858 | "5" | "6" | "7" | "8" | "9"
1859 UpperAlpha ::= "A" | "B" | "C" | "D" | "E" | "F" | "G"
1860 | "H" | "I" | "J" | "K" | "L" | "M" | "N"
1861 | "O" | "P" | "Q" | "R" | "S" | "T" | "U"
1862 | "V" | "W" | "X" | "Y" | "Z"
1863 LowerAlpha ::= "a" | "b" | "c" | "d" | "e" | "f" | "g"
1864 | "h" | "i" | "j" | "k" | "l" | "m" | "n"
1865 | "o" | "p" | "q" | "r" | "s" | "t" | "u"
1866 | "v" | "w" | "x" | "y" | "z"
1867 OtherChar ::= "!" | "'" | "(" | ")" | "*" | "+" | "," | "-"
1868 | "." | ":" | ";" | "=" | "_"
1869 UpperHexChar ::= Digit | "A" | "B" | "C" | "D" | "E" | "F"
1870 HexComponent ::= UpperHexChar+
1871 Escape ::= "%" HexChar HexChar
1872 HexChar ::= UpperHexChar | "a" | "b" | "c" | "d" | "e" | "f"
1873 GS3A3Char ::= Digit | UpperAlpha | LowerAlpha | OtherChar
1874 | Escape
1875 GS3A3Component ::= GS3A3Char+

1876 The syntactic construct GS3A3Component is used to represent fields of EAN.UCC codes
1877 that permit alphanumeric and other characters as specified in Figure 3A3-1 of the EAN.UCC
1878 General Specifications (see Appendix G). Owing to restrictions on URN syntax as defined
1879 by [RFC2141], not all characters permitted in the EAN.UCC General Specifications may be
1880 represented directly in a URN. Specifically, the characters " (double quote), % (percent), &
1881 (ampersand), / (forward slash), < (less than), > (greater than), and ? (question mark) are
1882 permitted in the General Specifications but may not be included directly in a URN. To
1883 represent one of these characters in a URN, escape notation must be used in which the
1884 character is represented by a percent sign, followed by two hexadecimal digits that give the
1885 ASCII character code for the character.

1886 **3.3.2 EPCGID-URI**

1887 EPCGID-URI ::= "urn:epc:id:gid:" 2*(NumericComponent ".")
1888 NumericComponent

1889 **3.3.3 SGTIN-URI**

1890 SGTIN-URI ::= "urn:epc:id:sgtin:" SGTINURIBody

1891 SGTINURIBody ::= 2*(PaddedNumericComponent ".") GS3A3Component

1892 The number of characters in the two PaddedNumericComponent fields must total 13
1893 (not including any of the dot characters).

1894 The Serial Number field of the SGTIN-URI is expressed as a GS3A3Component, which
1895 permits the representation of all characters permitted in the EAN.UCC-128 Application
1896 Identifier 21 Serial Number according to the EAN.UCC General Specifications. SGTIN-
1897 URIs that are derived from 96-bit tag encodings, however, will have Serial Numbers that
1898 consist only of digits and which have no leading zeros. These limitations are described in
1899 the encoding procedures, and in Appendix F.

1900 **3.3.4 SSCC-URI**

1901 SSCC-URI ::= "urn:epc:id:sscc:" SSCCURIBody

1902 SSCCURIBody ::= PaddedNumericComponent "."

1903 PaddedNumericComponent

1904 The number of characters in the two PaddedNumericComponent fields must total 17
1905 (not including any of the dot characters).

1906 **3.3.5 SGLN-URI**

1907 SGLN-URI ::= "urn:epc:id:sgln:" SGLNURIBody

1908 SGLNURIBody ::= 2*(PaddedNumericComponent ".") GS3A3Component

1909 The number of characters in the two PaddedNumericComponent fields must total 12
1910 (not including any of the dot characters).

1911 The *GLN Extension Component* field of the SGLN-URI is expressed as a
1912 GS3A3Component, which permits the representation of all characters permitted in the
1913 EAN.UCC-128 Application Identifier 254 Extension Component according to the EAN.UCC
1914 General Specifications. SGLN-URIs that are derived from 96-bit tag encodings, however,
1915 will have Extension Component that consist only of digits and which have no leading zeros.
1916 These limitations are described in the encoding procedures, and in Appendix F

1917 **3.3.6 GRAI-URI**

1918 GRAI-URI ::= "urn:epc:id:grai:" GRAIURIBody

1919 GRAIURIBody ::= 2*(PaddedNumericComponent ".") GS3A3Component

1920 The number of characters in the two `PaddedNumericComponent` fields must total 12
1921 (not including any of the dot characters).
1922 The Serial Number field of the GRAI-URI is expressed as a `GS3A3Component`, which
1923 permits the representation of all characters permitted in the Serial Number field of the GRAI
1924 according to the EAN.UCC General Specifications. GRAI-URIs that are derived from 96-bit
1925 tag encodings, however, will have Serial Numbers that consist only of digit characters and
1926 which have no leading zeros. These limitations are described in the encoding procedures,
1927 and in Appendix F.

1928 **3.3.7 GIAI-URI**

1929 `GIAI-URI ::= "urn:epc:id:giai:" GIAIURIBody`

1930 `GIAIURIBody ::= PaddedNumericComponent "." GS3A3Component`

1931 The total number of characters in the `PaddedNumericComponent` and
1932 `GS3A3Component` fields must not exceed 30 (not including the dot character that separates
1933 the two fields).

1934 The Individual Asset Reference field of the GIAI-URI is expressed as a `GS3A3Component`,
1935 which permits the representation of all characters permitted in the Individual Asset
1936 Reference field of the GIAI according to the EAN.UCC General Specifications. GIAI-URIs
1937 that is derived from 96-bit tag encodings, however, will have Individual Asset References
1938 that consist only of digit characters and which have no leading zeros. These limitations are
1939 described in the encoding procedures, and in Appendix F.

1940 **3.3.8 EPC Tag URI**

1941 `TagURI ::= "urn:epc:tag:" TagURIBody`

1942 `TagURIBody ::= GIDTagURIBody | SGTINSGLNGRAI96TagURIBody |`
1943 `SGTINSGLNGRAIAlphaTagURIBody | SSCCTagURIBody |`
1944 `GIAI96TagURIBody | GIAIAlphaTagURIBody`

1945 `GIDTagURIBody ::= GIDTagEncName ":" 2*(NumericComponent ".")`
1946 `NumericComponent`

1947 `GIDTagEncName ::= "gid-96"`

1948 `SGTINSGLNGRAITag96URIBody ::= SGTINSGLNGRAI96TagEncName ":"`
1949 `NumericComponent "." 2*(PaddedNumericComponent ".")`
1950 `NumericComponent`

1951 `SGTINSGLNGRAITagAlphaURIBody ::= SGTINSGLNGRAIAlphaTagEncName`
1952 `":" NumericComponent "." 2*(PaddedNumericComponent ".")`
1953 `GS3A3Component`

1954 `SGTINSGLNGRAI96TagEncName ::= "sgtin-96" | "sgln-96" | "grai-`
1955 `96"`

1956 `SGTINSGLNGRAIAlphaTagEncName ::= "sgtin-198" | "sgln-195" |`
1957 `"grai-170"`

1958 SSCCTagURIBody ::= SSCCTagEncName ":" NumericComponent 2*("."
1959 PaddedNumericComponent)
1960 SSCCTagEncName ::= "sscc-96"
1961 GIAI96TagURIBody ::= GIAI96TagEncName ":" NumericComponent "."
1962 PaddedNumericComponent "." NumericComponent
1963 GIAIAlphaTagURIBody ::= GIAIAlphaTagEncName ":"
1964 NumericComponent "." PaddedNumericComponent "." GS3A3Component
1965 GIAI96TagEncName ::= "giai-96"
1966 GIAIAlphaTagEncName ::= "giai-202"

1967 **3.3.9 Raw Tag URI**

1968 RawURI ::= "urn:epc:raw:" RawURIBody
1969 RawURIBody ::= DecimalRawURIBody | HexRawURIBody |
1970 AFIRawURIBody)
1971 DecimalRawURIBody ::= NonZeroComponent "." NumericComponent
1972 HexRawURIBody ::= NonZeroComponent ".x" HexComponent
1973 AFIRawURIBody ::= NonZeroComponent ".x" HexComponent ".x"
1974 HexComponent

1975 **3.3.10 EPC Pattern URI**

1976 PatURI ::= "urn:epc:pat:" PatBody
1977 PatBody ::= GIDPatURIBody | SGTINSGLNGRAI96PatURIBody |
1978 SGTINSGLNGRAIAlphaPatURIBody | SSCCPatURIBody |
1979 GIAI96PatURIBody | GIAIAlphaPatURIBody
1980 GIDPatURIBody ::= GIDTagEncName ":" 2*(PatComponent ".")
1981 PatComponent
1982 SGTINSGLNGRAI96PatURIBody ::= SGTINSGLNGRAI96TagEncName ":"
1983 PatComponent "." GS1PatBody "." PatComponent
1984 SGTINSGLNGRAIAlphaPatURIBody ::= SGTINSGLNGRAIAlphaTagEncName
1985 ":" PatComponent "." GS1PatBody "." GS3A3PatComponent
1986 SSCCPatURIBody ::= SSCCTagEncName ":" PatComponent "."
1987 GS1PatBody
1988 GIAI96PatURIBody ::= GIAI96TagEncName ":" PatComponent "."
1989 GS1PatBody
1990 GIAIAlphaPatURIBody ::= GIAIAlphaTagEncName ":" PatComponent
1991 "." GS1GS3A3PatBody
1992 GS1PatBody ::= "*.*" | (PaddedNumericComponent "."
1993 PatComponent)

1994 GS1GS3A3PatBody ::= "*" | (PaddedNumericComponent "."
1995 GS3A3PatComponent)
1996 PatComponent ::= NumericComponent
1997 | StarComponent
1998 | RangeComponent
1999 GS3A3PatComponent ::= GS3A3Component | StarComponent
2000 StarComponent ::= "*"
2001 RangeComponent ::= "[" NumericComponent "-"
2002 NumericComponent "]"
2003 For a RangeComponent to be legal, the numeric value of the first NumericComponent
2004 must be less than or equal to the numeric value of the second NumericComponent.

2005 **3.3.11 EPC Identity Pattern URI**

2006 IDPatURI ::= "urn:epc:idpat:" IDPatBody
2007 IDPatBody ::= GIDIDPatURIBody | SGTINIDPatURIBody |
2008 SGLNIDPatURIBody | GIAIIDPatURIBody | SSCCIDPatURIBody |
2009 GRAIIDPatURIBody
2010 GIDIDPatURIBody ::= "gid:" GIDIDPatURIMain
2011 GIDIDPatURIMain ::=
2012 2*(NumericComponent ".") NumericComponent
2013 | 2*(NumericComponent ".") "*"
2014 | NumericComponent ".*.*"
2015 | ".*.*"
2016 SGTINIDPatURIBody ::= "sgtin:" SGTINSGLNGRAIIDPatURIMain
2017 GRAIIDPatURIBody ::= "grai:" SGTINSGLNGRAIIDPatURIMain
2018 SGLNIDPatURIBody ::= "sgln:" SGTINSGLNGRAIIDPatURIMain
2019 SGTINSGLNGRAIIDPatURIMain ::=
2020 2*(PaddedNumericComponent ".") GS3A3Component
2021 | 2*(PaddedNumericComponent ".") "*"
2022 | PaddedNumericComponent ".*.*"
2023 | ".*.*"
2024 SSCCIDPatURIBody ::= "sscc:" SSCCIDPatURIMain
2025 SSCCIDPatURIMain ::=
2026 PaddedNumericComponent "." PaddedNumericComponent
2027 | PaddedNumericComponent ".*"
2028 | ".*"
2029 GIAIIDPatURIBody ::= "giai:" GIAIIDPatURIMain
2030 GIAIIDPatURIMain ::=
2031 PaddedNumericComponent "." GS3A3Component

2032 | PaddedNumericComponent `\".*\"`
2033 | `\"*. *\"`

2034 **3.3.12 DoD Construct URI**

2035 DOD-URI ::= `\"urn:epc:id:usdod:\" CAGECodeOrDODAAC \".\"`
2036 DoDSerialNumber

2037 DODTagURI ::= `\"urn:epc:tag:\" DoDTagType \":\" DoDFilter \".\"`
2038 CAGECodeOrDODAAC `\".\"` DoDSerialNumber

2039 DODPatURI ::= `\"urn:epc:pat:\" DoDTagType \":\" DoDFilterPat \".\"`
2040 CAGECodeOrDODAACPat `\".\"` DoDSerialNumberPat

2041 DODIDPatURI ::= `\"urn:epc:idpat:usdod:\" DODIDPatMain`

2042 DODIDPatMain ::=
2043 CAGECodeOrDODAAC `\".\"` DoDSerialNumber
2044 | CAGECodeOrDODAAC `\".*\"`
2045 | `\"*. *\"`

2046 DoDTagType ::= `\"usdod-96\"`

2047 DoDFilter ::= NumericComponent

2048 CAGECodeOrDODAAC ::= CAGECode | DODAAC

2049 CAGECode ::= CAGECodeOrDODAACChar*5

2050 DODAAC ::= CAGECodeOrDODAACChar*6

2051 DoDSerialNumber ::= NumericComponent

2052 DoDFilterPat ::= PatComponent

2053 CAGECodeOrDODAACPat ::= CAGECodeOrDODAAC | StarComponent

2054 DoDSerialNumberPat ::= PatComponent

2055 CAGECodeOrDODAACChar ::= Digit | `\"A\"` | `\"B\"` | `\"C\"` | `\"D\"` | `\"E\"` |
2056 `\"F\"` | `\"G\"` | `\"H\"` | `\"J\"` | `\"K\"` | `\"L\"` | `\"M\"` | `\"N\"` | `\"P\"` | `\"Q\"` |
2057 `\"R\"` | `\"S\"` | `\"T\"` | `\"U\"` | `\"V\"` | `\"W\"` | `\"X\"` | `\"Y\"` | `\"Z\"`

2058

2059 **3.3.13 Summary (non-normative)**

2060 The syntax rules above can be summarized informally as follows:

2061 `urn:epc:id:gid:MMM.CCC.SSS`

2062 `urn:epc:id:sgtin:PPP.III.AAA`

2063 `urn:epc:id:sscc:PPP.III`

2064 `urn:epc:id:sgln:PPP.III.AAA`

2065 `urn:epc:id:grai:PPP.III.AAA`

2066 urn:epc:id:giai:PPP.AAA
2067 urn:epc:id:usdod:TTT.SSS
2068
2069 urn:epc:tag:gid-96:MMM.CCC.SSS
2070 urn:epc:tag:sgtin-96:FFF.PPP.III.SSS
2071 urn:epc:tag:sgtin-198:FFF.PPP.III.AAA
2072 urn:epc:tag:sscc-96:FFF.PPP.III
2073 urn:epc:tag:sgln-96:FFF.PPP.III.SSS
2074 urn:epc:tag:sgln-195:FFF.PPP.III.AAA
2075 urn:epc:tag:grai-96:FFF.PPP.III.SSS
2076 urn:epc:tag:grai-170:FFF.PPP.III.AAA
2077 urn:epc:tag:giai-96:FFF.PPP.SSS
2078 urn:epc:tag:giai-202:FFF.PPP.AAA
2079 urn:epc:tag:usdod-96:FFF.TTT.SSS
2080
2081 urn:epc:raw:LLL.BBB
2082 urn:epc:raw:LLL.HHH
2083 urn:epc:raw:LLL.HHH.HHH
2084
2085 urn:epc:idpat:gid:MMM.CCC.SSS
2086 urn:epc:idpat:gid:MMM.CCC.*
2087 urn:epc:idpat:gid:MMM.*.*
2088 urn:epc:idpat:gid:*. *.*
2089 urn:epc:idpat:sgtin:PPP.III.AAA
2090 urn:epc:idpat:sgtin:PPP.III.*
2091 urn:epc:idpat:sgtin:PPP.*.*
2092 urn:epc:idpat:sgtin:*. *.*
2093 urn:epc:idpat:sscc:PPP.III
2094 urn:epc:idpat:sscc:PPP.*
2095 urn:epc:idpat:sscc:*. *
2096 urn:epc:idpat:sgln:PPP.III.AAA
2097 urn:epc:idpat:sgln:PPP.III.*
2098 urn:epc:idpat:sgln:PPP.*.*

2099 urn:epc:idpat:sgln:*. *.*
2100 urn:epc:idpat:grai:PPP.III.AAA
2101 urn:epc:idpat:grai:PPP.III.*
2102 urn:epc:idpat:grai:PPP.*.*
2103 urn:epc:idpat:grai:*. *.*
2104 urn:epc:idpat:giai:PPP.AAA
2105 urn:epc:idpat:giai:PPP.*
2106 urn:epc:idpat:giai:*. *
2107 urn:epc:idpat:usdod:TTT.SSS
2108
2109 urn:epc:idpat:usdod:TTT.*
2110 urn:epc:idpat:usdod:*. *
2111
2112 urn:epc:pat:gid-96:MMMpat.CCCpat.SSSpat
2113 urn:epc:pat:sgtin-96:FFFpat.PPP.IIIpat.SSSpat
2114 urn:epc:pat:sgtin-96:FFFpat.*.*.SSSpat
2115 urn:epc:pat:sgtin-198:FFFpat.PPP.IIIpat.AAApat
2116 urn:epc:pat:sgtin-198:FFFpat.*.*.AAAp
2117 urn:epc:pat:sscc-96:FFFpat.PPP.IIIpat
2118 urn:epc:pat:sscc-96:FFFpat.*.*
2119 urn:epc:pat:sgln-96:FFFpat.PPP.IIIpat.SSSpat
2120 urn:epc:pat:sgln-96:FFFpat.*.*.SSSpat
2121 urn:epc:pat:sgln-195:FFFpat.PPP.IIIpat.AAApat
2122 urn:epc:pat:sgln-195:FFFpat.*.*.AAAp
2123 urn:epc:pat:grai-96:FFFpat.PPP.IIIpat.SSSpat
2124 urn:epc:pat:grai-96:FFFpat.*.*.SSSpat
2125 urn:epc:pat:grai-170:FFFpat.PPP.IIIpat.AAApat
2126 urn:epc:pat:grai-170:FFFpat.*.*.AAAp
2127 urn:epc:pat:giai-96:FFFpat.PPP.SSSpat
2128 urn:epc:pat:giai-96:FFFpat.*.*
2129 urn:epc:pat:giai-202:FFFpat.PPP.AAApat
2130 urn:epc:pat:giai-202:FFFpat.*.*
2131 urn:epc:pat:usdod-96:FFFpat.TTT.SSSpat

2132 urn:epc:pat:usdod-96:FFFpat.*.SSSpat
 2133 where
 2134 *MMM* denotes a General Manager Number
 2135 *CCC* denotes an Object Class number
 2136 *SSS* denotes a numeric Serial Number or GIAI Individual Asset Reference
 2137 *AAA* denotes an alphanumeric Serial Number or GIAI Individual Asset reference
 2138 *PPP* denotes an EAN.UCC Company Prefix
 2139 *TTT* denotes a US DoD assigned CAGE code or DODAAC
 2140 *III* denotes an SGTIN Item Reference (prefixed by the Indicator Digit), an SSCC
 2141 Shipping Container Serial Number (prefixed by the Extension Digit (ED)), a SGLN Location
 2142 Reference, or a GRAI Asset Type.
 2143 *FFF* denotes a filter code as used by the SGTIN, SSCC, SGLN, GRAI, GIAI, and DoD tag
 2144 encodings
 2145 *XXXpat* is the same as *XXX* but allowing * and [lo-hi] pattern syntax in addition
 2146 (exception: [lo-hi] syntax is not allowed for *AAAp*at).
 2147 *LLL* denotes the number of bits of an uninterpreted bit sequence
 2148 *BBB* denotes the literal value of an uninterpreted bit sequence converted to decimal
 2149 *HHH* denotes the literal value of an uninterpreted bit sequence converted to hexadecimal
 2150 and preceded by the character 'x'.
 2151 and where all numeric fields are in decimal with no leading zeros (unless the overall value of
 2152 the field is zero, in which case it is represented with a single 0 character), with the exception
 2153 of the hexadecimal raw representation.
 2154 Exceptions:
 2155 1. The length of *PPP* and *III* is significant, and leading zeros are used as necessary.
 2156 The length of *PPP* is the length of the company prefix as assigned by GS1. The
 2157 length of *III* plus the length of *PPP* must equal 13 for SGTIN, 17 for SSCC, 12 for
 2158 GLN, or 12 for GRAI.
 2159 2. The Value field of urn:epc:raw is expressed in hexadecimal if the value is
 2160 preceded by the character 'x'.

2161 **4 Translation between EPC-URI and Other EPC** 2162 **Representations**

2163 This section defines the semantics of EPC-URI encodings, by defining how they are
 2164 translated into other EPC representations and vice versa.

2165 **4.1 Bit string into EPC-URI (pure identity)**

2166 The following procedure translates a bit-level encoding into an EPC-URI:

- 2167 1. Determine the identity type and encoding scheme by finding the row in Table 1
2168 (Section 2.1) that matches the most significant bits of the bit string. If the most
2169 significant bits do not match any row of the table, stop: the bit string is invalid and
2170 cannot be translated into an EPC-URI. If the encoding scheme indicates one of the
2171 DoD Tag Data Constructs, consult the appropriate U.S. Department of Defense
2172 document for specific encoding and decoding rules. Otherwise, if the encoding
2173 scheme is SGTIN-96 or SGTIN-198, proceed to Step 2; if the encoding scheme is
2174 SSCC-96, proceed to Step 5; if the encoding scheme is SGLN-96 or SGLN-195,
2175 proceed to Step 8; if the encoding scheme is GRAI-96 or GRAI-170, proceed to
2176 Step 11; if the encoding scheme is GIAI-96 or GIAI-202, proceed to Step 14; if the
2177 encoding scheme is GID-96, proceed to Step 17.
- 2178 2. Follow the decoding procedure given in Section 3.5.1.2 (for SGTIN-96) or in
2179 Section 3.5.2.2 (for SGTIN-198) to obtain the decimal Company Prefix $p_1p_2...p_L$, the
2180 decimal Item Reference and Indicator $i_1i_2...i_{(13-L)}$, and the Serial Number S . If the
2181 decoding procedure fails, stop: the bit-level encoding cannot be translated into an
2182 EPC-URI.
- 2183 3. Create an EPC-URI by concatenating the following: the string
2184 `urn:epc:id:sgtin:`, the Company Prefix $p_1p_2...p_L$ where each digit (including
2185 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
2186 character, the Item Reference and Indicator $i_1i_2...i_{(13-L)}$ (handled similarly), a dot (.)
2187 character, and the Serial Number S as a decimal integer (SGTIN-96) or alphanumeric
2188 character (SGTIN-198). For SGTIN-96 the portion corresponding to the Serial
2189 Number must have no leading zeros, except where the Serial Number is itself zero in
2190 which case the corresponding URI portion must consist of a single zero character.
- 2191 4. Go to Step 19.
- 2192 5. Follow the decoding procedure given in Section 3.6.1.2 (for SSCC-96) to obtain the
2193 decimal Company Prefix $p_1p_2...p_L$, and the decimal Serial Reference $s_1s_2...s_{(17-L)}$. If
2194 the decoding procedure fails, stop: the bit-level encoding cannot be translated into an
2195 EPC-URI.
- 2196 6. Create an EPC-URI by concatenating the following: the string
2197 `urn:epc:id:sscc:`, the Company Prefix $p_1p_2...p_L$ where each digit (including
2198 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
2199 character, and the Serial Reference $s_1s_2...s_{(17-L)}$ (handled similarly).
- 2200 7. Go to Step 19.
- 2201 8. Follow the decoding procedure given in Section 3.7.1.2 (for SGLN-96) or in Section
2202 3.7.2.2 (for SGLN-195) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal
2203 Location Reference $i_1i_2...i_{(12-L)}$, and the Extension Component S . If the decoding
2204 procedure fails, stop: the bit-level encoding cannot be translated into an EPC-URI.

- 2205 9. Create an EPC-URI by concatenating the following: the string
 2206 urn:epc:id:sgln:, the Company Prefix $p_1p_2...p_L$ where each digit (including
 2207 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
 2208 character, for $L < 12$ the Location Reference, $i_1i_2...i_{(12-L)}$ (handled similarly), a dot
 2209 (.) character, and the Extension Component S as a decimal integer (SGLN-96) or
 2210 alphanumeric character (SGLN-195). For SGLN-96 the portion corresponding to the
 2211 Extension Component must have no leading zeros, except where the Extension
 2212 Component is itself zero in which case the corresponding URI portion must consist of
 2213 a single zero character. If a Location Reference does not exist (where $L = 12$), leave
 2214 no blank space between the two dot (.) characters.
- 2215 10. Go to Step 19.
- 2216 11. Follow the decoding procedure given in Section 3.8.1.2 (for GRAI-96) or in Section
 2217 3.8.2.2 (for GRAI-170) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal
 2218 Asset Type $i_1i_2...i_{(12-L)}$, and the Serial Number S . If the decoding procedure fails,
 2219 stop: the bit-level encoding cannot be translated into an EPC-URI.
- 2220 12. Create an EPC-URI by concatenating the following: the string
 2221 urn:epc:id:grai:, the Company Prefix $p_1p_2...p_L$ where each digit (including
 2222 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
 2223 character, for $L < 12$ the Asset Type $i_1i_2...i_{(12-L)}$ (handled similarly), a dot (.)
 2224 character, and the Serial Number S as a decimal integer (GRAI-96) or alphanumeric
 2225 character (GRAI-170). For GRAI-96 the portion corresponding to the Serial Number
 2226 must have no leading zeros, except where the Serial Number is itself zero in which
 2227 case the corresponding URI portion must consist of a single zero character. If an
 2228 Asset Type does not exist (where $L = 12$), leave no blank space between the two dot
 2229 (.) characters.
- 2230 13. Go to Step 19.
- 2231 14. Follow the decoding procedure given in Section 3.9.1.2 (for GIAI-96) or in 3.9.2.2
 2232 (for GIAI-202) to obtain the decimal Company Prefix $p_1p_2...p_L$, and the Individual
 2233 Asset Reference S . If the decoding procedure fails, stop: the bit-level encoding
 2234 cannot be translated into an EPC-URI.
- 2235 15. Create an EPC-URI by concatenating the following: the string
 2236 urn:epc:id:giai:, the Company Prefix $p_1p_2...p_L$ where each digit (including
 2237 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)
 2238 character, and the Individual Asset Reference S as a decimal integer (GIAI-96) or
 2239 alphanumeric character (GIAI-202). For GIAI-96 the portion corresponding to the
 2240 Individual Asset Reference must have no leading zeros, except where the Individual
 2241 Asset Reference is itself zero in which case the corresponding URI portion must
 2242 consist of a single zero character.
- 2243 16. Go to Step 19.
- 2244 17. Follow the decoding procedure given in Section 3.4.1.2 to obtain the General
 2245 Manager Number M , the Object Class C , and the Serial Number S .

- 2246 18. Create an EPC-URI by concatenating the following: the string `urn:epc:id:gid:`,
 2247 the General Manager Number as a decimal integer, a dot (.) character, the Object
 2248 Class as a decimal integer, a dot (.) character, and the Serial Number S as a decimal
 2249 integer. Each decimal number must have no leading zeros, except where the integer
 2250 is itself zero in which case the corresponding URI portion must consist of a single
 2251 zero character.
- 2252 19. The translation is now complete.

2253 4.2 Bit String into Tag or Raw URI

2254 The following procedure translates a bit string of N bits into either an EPC Tag URI or a
 2255 Raw Tag URI:

- 2256 1. Determine the identity type, encoding scheme, and encoding length (K) by finding
 2257 the row in Table 1 (Section 2.1) that matches the most significant bits of the bit string.
 2258 If $N < K$, proceed to Step 20; otherwise, continue with the remainder of this
 2259 procedure, using the most significant K bits of the bit string. If the encoding scheme
 2260 indicates one of the DoD Tag Data Constructs, consult the appropriate U.S.
 2261 Department of Defense document for specific encoding and decoding rules. If the
 2262 encoding scheme is SGTIN-96 or SGTIN-198, proceed to Step 2; if the encoding
 2263 scheme is SSCC-96, proceed to Step 5; if the encoding scheme is SGLN-96 or
 2264 SGLN-195, proceed to Step 8; if the encoding scheme is GRAI-96 or GRAI-170,
 2265 proceed to Step 11, if the encoding scheme is GIAI-96 or GIAI-202, proceed to Step
 2266 14, if the encoding scheme is GID-96, proceed to Step 17; otherwise, proceed to Step
 2267 20.
- 2268 2. Follow the decoding procedure given in Section 3.5.1.2 (for SGTIN-96) or 3.5.2.2
 2269 (for SGTIN-198) to obtain the decimal Company Prefix $p_1p_2\dots p_L$, the decimal Item
 2270 Reference and Indicator $i_1i_2\dots i_{(13-L)}$, the Filter Value F , and the Serial Number S . If
 2271 the decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2272 3. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,
 2273 the encoding scheme (`sgtin-96` or `sgtin-198`), a colon (:) character, the Filter
 2274 Value F as a decimal integer, a dot (.) character, the Company Prefix $p_1p_2\dots p_L$ where
 2275 each digit (including any leading zeros) becomes the corresponding ASCII digit
 2276 character, a dot (.) character, the Item Reference and Indicator $i_1i_2\dots i_{(13-L)}$ (handled
 2277 similarly), a dot (.) character, and the Serial Number S as a decimal integer (SGTIN-
 2278 96) or alphanumeric character (SGTIN-198). For SGTIN-96 the portions
 2279 corresponding to the Filter Value and Serial Number must have no leading zeros,
 2280 except where the corresponding integer is itself zero in which case a single zero
 2281 character is used.
- 2282 4. Go to Step 21.
- 2283 5. Follow the decoding procedure given in Section 3.6.1.2 (for SSCC-96) to obtain the
 2284 decimal Company Prefix $p_1p_2\dots p_L$, and the decimal Serial Reference $i_1i_2\dots i_{(17-L)}$, and
 2285 the Filter Value F . If the decoding procedure fails, proceed to Step 20, otherwise
 2286 proceed to the next step.

- 2287 6. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,
 2288 the encoding scheme (`sscc-96`), a colon (`:`) character, the Filter Value F as a
 2289 decimal integer, a dot (`.`) character, the Company Prefix $p_1p_2...p_L$ where each digit
 2290 (including any leading zeros) becomes the corresponding ASCII digit character, a dot
 2291 (`.`) character, and the Serial Reference $i_1i_2...i_{(17-L)}$ (handled similarly).
- 2292 7. Go to Step 21.
- 2293 8. Follow the decoding procedure given in Section 3.7.1.2 (for SGLN-96) or Section
 2294 3.7.2.2 (for SGLN-195) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal
 2295 Location Reference $i_1i_2...i_{(12-L)}$, the Filter Value F , and the Extension Component S .
 2296 If the decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2297 9. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,
 2298 the encoding scheme (`s9ln-96` or `s9ln-195`), a colon (`:`) character, the Filter
 2299 Value F as a decimal integer, a dot (`.`) character, the Company Prefix $p_1p_2...p_L$ where
 2300 each digit (including any leading zeros) becomes the corresponding ASCII digit
 2301 character, a dot (`.`) character, when $L < 12$ the Location Reference $i_1i_2...i_{(12-L)}$
 2302 (handled similarly), a dot (`.`) character, and the Extension Component S as a decimal
 2303 integer (SGLN-96) or alphanumeric character (SGLN-198). For SGLN-96 the
 2304 portions corresponding to the Filter Value and Extension Component must have no
 2305 leading zeros, except where the corresponding integer is itself zero in which case a
 2306 single zero character is used. If a Location Reference does not exist where $L = 12$
 2307 leave no blank space between the two dot (`.`) characters.
- 2308 10. Go to Step 21.
- 2309 11. Follow the decoding procedure given in Section 3.8.1.2 (for GRAI-96) or 3.8.2.2 (for
 2310 GRAI-170) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal Asset Type
 2311 $i_1i_2...i_{(12-L)}$, the Filter Value F , and the Serial Number $d_1d_2...d_K$. If the decoding
 2312 procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2313 12. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,
 2314 the encoding scheme (`grai-96` or `grai-170`), a colon (`:`) character, the Filter
 2315 Value F as a decimal integer, a dot (`.`) character, the Company Prefix $p_1p_2...p_L$ where
 2316 each digit (including any leading zeros) becomes the corresponding ASCII digit
 2317 character, a dot (`.`) character, for $L < 12$ the Asset Type $i_1i_2...i_{(12-L)}$ (handled
 2318 similarly), a dot (`.`) character, and the Serial Number $d_1d_2...d_K$ as a decimal integer
 2319 (GRAI-96) or alphanumeric character (GRAI-170). For GRAI-96 the portions
 2320 corresponding to the Filter Value and Serial Number must have no leading zeros,
 2321 except where the corresponding integer is itself zero in which case a single zero
 2322 character is used. If an Asset Type does not exist where $L = 12$ leave no blank space
 2323 between the two dot (`.`) characters.
- 2324 13. Got to Step 21.
- 2325 14. Follow the decoding procedure given in Section 3.9.1.2 (for GIAI-96) or 3.9.2.2 (for
 2326 GIAI-202) to obtain the decimal Company Prefix $p_1p_2...p_L$, the Individual Asset
 2327 Reference $s_1s_2...s_J$, and the Filter Value F . If the decoding procedure fails, proceed
 2328 to Step 20, otherwise proceed to the next step.

- 2329 15. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,
 2330 the encoding scheme (`giai-96` or `giai-202`), a colon (`:`) character, the Filter
 2331 Value F as a decimal integer, a dot (`.`) character, the Company Prefix $p_1p_2...p_L$ where
 2332 each digit (including any leading zeros) becomes the corresponding ASCII digit
 2333 character, a dot (`.`) character, and the Individual Asset Reference $s_1s_2...s_J$ (handled
 2334 similarly). For GIAI-96 the portion corresponding to the Filter Value and the
 2335 Individual Asset Reference must have no leading zeros, except where the
 2336 corresponding integer is itself zero in which case a single zero character is used.
- 2337 16. Go to Step 21.
- 2338 17. Follow the decoding procedure given in Section 3.4.1.2 to obtain the General
 2339 Manager Number, the Object Class, and the Serial Number.
- 2340 18. Create an EPC Tag URI by concatenating the following: the string
 2341 `urn:epc:tag:gid-96:`, the General Manager Number as a decimal number, a
 2342 dot (`.`) character, the Object Class as a decimal number, a dot (`.`) character, and the
 2343 Serial Number as a decimal number. Each decimal number must have no leading
 2344 zeros, except where the integer is itself zero in which case the corresponding URI
 2345 portion must consist of a single zero character.
- 2346 19. Go to Step 21.
- 2347 20. This tag is not a recognized EPC Tag Encoding, therefore create an EPC Raw URI by
 2348 concatenating the following: the string `urn:epc:raw:`, the length of the bit string
 2349 (N) expressed as a decimal integer with no leading zeros, a dot (`.`) character, a
 2350 lowercase `x` character, and the value of the bit string considered as a single
 2351 hexadecimal integer. The value must have a number of characters equal to the length
 2352 (N) divided by four and rounded up to the nearest whole number, and must only use
 2353 uppercase letters for the hexadecimal digits A, B, C, D, E, and F.
- 2354 21. The translation is now complete.
- 2355

2356 **4.3 Gen 2 Tag EPC Memory into EPC-URI (pure identity)**

2357 The following procedure translates the contents of the EPC Memory of a Gen 2 Tag into an
 2358 EPC-URI:

- 2359 1. Consider bits $10x$ through $14x$ (inclusive) as a five-bit binary integer, L .
- 2360 2. Examine the “toggle” bit, bit $17x$. If the toggle bit is a one, stop: this bit string
 2361 cannot be converted into an EPC-URI. Otherwise, continue with Step 3.
- 2362 3. Extract N bits beginning with bit $20x$, where $N = 16L$.
- 2363 4. Finish by proceeding with the procedure in Section 4.1, using the N -bit string
 2364 extracted in Step 3.

2365 **4.4 Gen 2 Tag EPC Memory into Tag or Raw URI**

2366 The following procedure translates the contents of the EPC Memory of a Gen 2 Tag into
2367 either an EPC Tag URI or a Raw Tag URI:

- 2368 1. Consider bits 10x through 14x (inclusive) as a five-bit binary integer, L.
- 2369 2. Examine the “toggle” bit, bit 17x. If the toggle bit is a one, go to Step 5. Otherwise,
2370 continue with Step 3.
- 2371 3. Extract N bits beginning with bit 20x, where N = 16L.
- 2372 4. Finish by proceeding with the procedure in Section 4.2, using the N-bit string
2373 extracted in Step 3.
- 2374 5. This tag has an AFI, and is therefore by definition not an EPC Tag Encoding.
2375 Continue with the following steps.
- 2376 6. Extract bits 18x through 1Fx (inclusive) as an eight-bit binary integer, A (this is the
2377 AFI).
- 2378 7. Extract N bits beginning with bit 20x, where N = 16L.
- 2379 8. Create an EPC Raw URI by concatenating the following: the string
2380 urn:epc:raw:, the number N from Step 7 expressed as a decimal integer with no
2381 leading zeros, a dot (.) character, a lowercase x character, the value A from Step 6
2382 expressed as a two-character hexadecimal integer, a dot (.) character, a lowercase x
2383 character, and the value of the N-bit string from Step 7 considered as a single
2384 hexadecimal integer. The value must have a number of characters equal to the length
2385 (N) divided by four. Both the AFI and the value must only use uppercase letters for
2386 the hexadecimal digits A, B, C, D, E, and F.

2387 **4.5 URI into Bit String**

2388 The following procedure translates a URI into a bit string:

- 2389 1. If the URI is an SGTIN-URI (urn:epc:id:sgtin:), an SSCC-URI
2390 (urn:epc:id:sscc:), an SGLN-URI (urn:epc:id:sgln:), a GRAI-URI
2391 (urn:epc:id:grai:), a GIAI-URI (urn:epc:id:giai:), a GID-URI
2392 (urn:epc:id:gid:), a DOD-URI (urn:epc:id:usdod:) or an EPC Pattern
2393 URI (urn:epc:pat:), the URI cannot be translated into a bit string.
- 2394 2. If the URI is a Raw Tag URI of the form urn:epc:raw:N.V, create the bit string
2395 by converting the second component (V) of the Raw Tag URI into a binary integer,
2396 whose length is equal to the first component (N) of the Raw Tag URI. If the value of
2397 the second component is too large to fit into a binary integer of that size, the URI
2398 cannot be translated into a bit string. If the URI is a Raw Tag URI of the form
2399 urn:epc:raw:N.A.V, the URI cannot be translated into a bit string (but see the
2400 related procedure in Section 4.6).
- 2401 3. If the URI is an EPC Tag URI or US DoD Tag URI (urn:epc:tag:encName:),
2402 parse the URI using the grammar for TagURI as given in Section 3.3.8 or for

2403 DODTagURI as given in Section 4.3.11. If the URI cannot be parsed using these
2404 grammars, stop: the URI is illegal and cannot be translated into a bit string. If
2405 *encName* is *usdod-96*, consult the appropriate U.S. Department of Defense
2406 document for specific translation rules. Otherwise, if *encName* is *sgtin-96* go to
2407 Step 4, if *sgtin-198* go to Step 9, if *encName* is *sscc-96* go to Step 14, if
2408 *encName* is *sgln-96* go to Step 18 or *sgln-195* go to Step 23, if *encName* is
2409 *grai-96* go to Step 28 or *grai-170* go to Step 33, if *encName* is *giai-96* go
2410 to Step 38 or *giai-202* go to Step 43, or if *encName* is *gid-96* go to Step 48.

2411 4. Let the URI be written as
2412 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(13-L)} \cdot s_1 s_2 \dots s_S$.

2413 5. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .

2414 6. Interpret $s_1 s_2 \dots s_S$ as a decimal integer S .

2415 7. Carry out the encoding procedure defined in Section 3.5.1.1 (SGTIN-96), using
2416 $i_1 p_1 p_2 \dots p_L i_2 \dots i_{(13-L)} 0$ as the EAN.UCC GTIN-14 (the trailing zero is a dummy
2417 check digit, which is ignored by the encoding procedure), L as the length of the
2418 EAN.UCC company prefix, F from Step 5 as the Filter Value, and S from Step 6 as
2419 the Serial Number. If the encoding procedure fails because an input is out of range,
2420 or because the procedure indicates a failure, stop: this URI cannot be encoded into a
2421 bit string.

2422 8. Go to Step 53.

2423 9. Let the URI be written as
2424 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(13-L)} \cdot s_1 s_2 \dots s_S$.

2425 10. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .

2426 11. Interpret $s_1 s_2 \dots s_S$ as an alphanumeric string S .

2427 12. Carry out the encoding procedure defined in Section 3.5.2.1 (SGTIN-198) using
2428 $i_1 p_1 p_2 \dots p_L i_2 \dots i_{(13-L)} 0$ as the EAN.UCC GTIN-14 (the trailing zero is a dummy
2429 check digit, which is ignored by the encoding procedure), L as the length of the
2430 EAN.UCC company prefix, F from Step 10 as the Filter Value, and S from Step 11
2431 as the Serial Number. If the encoding procedure fails because an input is out of range,
2432 or because the procedure indicates a failure, stop: this URI cannot be encoded into a
2433 bit string.

2434 13. Go to Step 53.

2435 14. Let the URI be written as
2436 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(17-L)}$.

2437 15. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .

2438 16. Carry out the encoding procedure defined in Section 3.6.1.1 (SSCC-96), using
2439 $i_1 p_1 p_2 \dots p_L i_2 i_3 \dots i_{(17-L)} 0$ as the EAN.UCC SSCC (the trailing zero is a dummy
2440 check digit, which is ignored by the encoding procedure), L as the length of the
2441 EAN.UCC company prefix, and F from Step 15 as the Filter Value. If the encoding

- 2442 procedure fails because an input is out of range, or because the procedure indicates a
 2443 failure, stop: this URI cannot be encoded into a bit string.
- 2444 17. Go to Step 53.
- 2445 18. Let the URI be written as
 2446 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(12-L)} \cdot s_1 s_2 \dots s_S$.
- 2447 19. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .
- 2448 20. Interpret $s_1 s_2 \dots s_S$ as a decimal integer S .
- 2449 21. Carry out the encoding procedure defined in Section 3.7.1.1 (SGLN-96), using
 2450 $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0$ as the EAN.UCC GLN (the trailing zero is a dummy check
 2451 digit, which is ignored by the encoding procedure), L as the length of the EAN.UCC
 2452 company prefix, F from Step 19 as the Filter Value, and S from Step 20 as the
 2453 Extension Component. If the encoding procedure fails because an input is out of
 2454 range, or because the procedure indicates a failure, stop: this URI cannot be encoded
 2455 into a bit string.
- 2456 22. Go to Step 53.
- 2457 23. Let the URI be written as
 2458 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(12-L)} \cdot s_1 s_2 \dots s_S$.
- 2459 24. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F .
- 2460 25. Interpret $s_1 s_2 \dots s_S$ as an alphanumeric string S .
- 2461 26. Carry out the encoding procedure defined in Section 3.7.2.1 (SGLN-195), using
 2462 $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0$ as the EAN.UCC GLN (the trailing zero is a dummy check
 2463 digit, which is ignored by the encoding procedure), L as the length of the EAN.UCC
 2464 company prefix, F from Step 24 as the Filter Value, and S from Step 25 as the
 2465 Extension Component. If the encoding procedure fails because an input is out of
 2466 range, or because the procedure indicates a failure, stop: this URI cannot be encoded
 2467 into a bit string.
- 2468 27. Go to Step 53.
- 2469 28. Let the URI be written as
 2470 $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(12-L)} \cdot s_1 s_2 \dots s_S$.
- 2471 29. Interpret $f_1 f_2 \dots f_F$ as a decimal integer F
- 2472 30. Interpret $s_1 s_2 \dots s_S$ as a decimal integer S .
- 2473 31. Carry out the encoding procedure defined in Section 3.8.1.1 (GRAI-96), using
 2474 $0 p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0 s_1 s_2 \dots s_S$ as the EAN.UCC GRAI (the second zero is a
 2475 dummy check digit, which is ignored by the encoding procedure), L as the length of
 2476 the EAN.UCC company prefix, and F from Step 29 as the Filter Value, and S from
 2477 Step 30 as the Serial Number. If the encoding procedure fails because an input is out
 2478 of range, or because the procedure indicates a failure, stop: this URI cannot be
 2479 encoded into a bit string.

- 2480 32. Go to Step 53.
- 2481 33. Let the URI be written as
- 2482 $\text{urn:epc:tag:encName:f}_1\text{f}_2\dots\text{f}_F \cdot \text{p}_1\text{p}_2\dots\text{p}_L \cdot \text{i}_1\text{i}_2\dots\text{i}_{(12-L)} \cdot \text{s}_1\text{s}_2\dots\text{s}_S$.
- 2483 34. Interpret $\text{f}_1\text{f}_2\dots\text{f}_F$ as a decimal integer F .
- 2484 35. Interpret $\text{s}_1\text{s}_2\dots\text{s}_S$ as an alphanumeric string S .
- 2485 36. Carry out the encoding procedure defined in Section 3.8.2.1 (GRAI-170) using
- 2486 $0\text{p}_1\text{p}_2\dots\text{p}_L\text{i}_1\text{i}_2\dots\text{i}_{(12-L)}0\text{s}_1\text{s}_2\dots\text{s}_S$ as the EAN.UCC GRAI (the second zero is a
- 2487 dummy check digit, which is ignored by the encoding procedure), L as the length of
- 2488 the EAN.UCC company prefix, and F from Step 34 as the Filter Value, and S from
- 2489 Step 35 as the Serial Number. If the encoding procedure fails because an input is out
- 2490 of range, or because the procedure indicates a failure, stop: this URI cannot be
- 2491 encoded into a bit string.
- 2492 37. Go to Step 53.
- 2493 38. Let the URI be written as $\text{urn:epc:tag:encName:f}_1\text{f}_2\dots\text{f}_F \cdot \text{p}_1\text{p}_2\dots\text{p}_L \cdot \text{s}_1\text{s}_2\dots\text{s}_S$.
- 2494 39. Interpret $\text{f}_1\text{f}_2\dots\text{f}_F$ as a decimal integer F
- 2495 40. Interpret $\text{s}_1\text{s}_2\dots\text{s}_S$ as a decimal integer S .
- 2496 41. Carry out the encoding procedure defined in Section 3.9.1.1 (GIAI-96), using
- 2497 $\text{p}_1\text{p}_2\dots\text{p}_L\text{s}_1\text{s}_2\dots\text{s}_S$ as the EAN.UCC GIAI, L as the length of the EAN.UCC company
- 2498 prefix, and F from Step 39 as the Filter Value, and S from Step 40 as the Serial
- 2499 Number. If the encoding procedure fails because an input is out of range, or because
- 2500 the procedure indicates a failure, stop: this URI cannot be encoded into a bit string.
- 2501 42. Go to Step 53.
- 2502 43. Let the URI be written as $\text{urn:epc:tag:encName:f}_1\text{f}_2\dots\text{f}_F \cdot \text{p}_1\text{p}_2\dots\text{p}_L \cdot \text{s}_1\text{s}_2\dots\text{s}_S$.
- 2503 44. Interpret $\text{f}_1\text{f}_2\dots\text{f}_F$ as a decimal integer F .
- 2504 45. Interpret $\text{s}_1\text{s}_2\dots\text{s}_S$ as an alphanumeric string S .
- 2505 46. Carry out the encoding procedure defined in Section 3.9.2.1 (GIAI-202) using
- 2506 $\text{p}_1\text{p}_2\dots\text{p}_L\text{s}_1\text{s}_2\dots\text{s}_S$ as the EAN.UCC GIAI, L as the length of the EAN.UCC company
- 2507 prefix, and F from Step 44 as the Filter Value, and S from Step 45 as the Serial
- 2508 Number. If the encoding procedure fails because an input is out of range, or because
- 2509 the procedure indicates a failure, stop: this URI cannot be encoded into a bit string.
- 2510 47. Go to Step 53.
- 2511 48. Let the URI be written as $\text{urn:epc:tag:encName:m}_1\text{m}_2\dots\text{m}_L \cdot \text{c}_1\text{c}_2\dots\text{c}_K \cdot \text{s}_1\text{s}_2\dots\text{s}_S$.
- 2512 49. Interpret $\text{m}_1\text{m}_2\dots\text{m}_L$ as a decimal integer M .
- 2513 50. Interpret $\text{c}_1\text{c}_2\dots\text{c}_K$ as a decimal integer C .
- 2514 51. Interpret $\text{s}_1\text{s}_2\dots\text{s}_S$ as a decimal integer S .

- 2515 52. Carry out the encoding procedure defined in Section 3.4.1.1 using M from Step 49 as
2516 the General Manager Number, C from Step 50 as the Object Class, and S from
2517 Step 51 as the Serial Number. If the encoding procedure fails because an input is out
2518 of range, or because the procedure indicates a failure, stop: this URI cannot be
2519 encoded into a bit string.
- 2520 53. The translation is complete.

2521 4.6 URI into Gen 2 Tag EPC Memory

2522 The following procedure converts a URI into a sequence of bits suitable for writing into the
2523 EPC memory of a Gen 2 Tag, starting with bit 10x (i.e., not including the CRC).

- 2524 1. If the URI is a Raw Tag URI of the form `urn:epc:raw:N.A.V`, calculate the
2525 value L , where $L = N/16$ rounded up to the nearest whole number. If $L \geq 32$, stop:
2526 this URI cannot be encoded into the EPC memory of a Gen 2 Tag. If $A \geq 256$ or if
2527 the value V is too large to be expressed as an N -bit binary integer, stop: this URI
2528 cannot be encoded into the EPC memory of a Gen 2 Tag. Otherwise, construct the
2529 contents of EPC memory by concatenating the following bit strings: the value L (five
2530 bits), two zero bits (00), a single one bit (1), the value A (eight bits), and the value V
2531 (16L bits).
- 2532 2. Otherwise, apply the procedure of Section 4.5 to obtain an N -bit string, V . If the
2533 procedure of Section 4.5 fails, stop: this URI cannot be encoded into the EPC
2534 memory of a Gen 2 Tag. Otherwise, calculate $L = N/16$ rounded up to the nearest
2535 whole number. Construct the contents of EPC memory by concatenating the
2536 following bit strings: the value L (five bits), eleven zero bits (00000000000), the
2537 value V (N bits), and as many zero bits as required to make a total of $16(L+1)$ bits.

2538 5 Semantics of EPC Pattern URIs

2539 The meaning of an EPC Pattern URI (`urn:epc:pat:`) or EPC Pure Identity Pattern URI
2540 (`urn:epc:idpat:`) can be formally defined as denoting a set of encoding-specific EPCs
2541 or a set of pure identity EPCs, respectively.

2542 The set of EPCs denoted by a specific EPC Pattern URI is defined by the following decision
2543 procedure, which says whether a given EPC Tag URI belongs to the set denoted by the EPC
2544 Pattern URI.

2545 Let `urn:epc:pat:EncName:P1.P2...Pn` be an EPC Pattern URI. Let
2546 `urn:epc:tag:EncName:C1.C2...Cn` be an EPC Tag URI, where the *EncName* field
2547 of both URIs is the same. The number of components (n) depends on the value of
2548 *EncName*.

2549 First, any EPC Tag URI component C_i is said to *match* the corresponding EPC Pattern URI
2550 component P_i if:

- 2551 • P_i is a `NumericComponent`, and C_i is equal to P_i ; or

- 2552 • P_i is a PaddedNumericComponent, and C_i is equal to P_i both in numeric value as
2553 well as in length; or
- 2554 • P_i is a GS3A3Component, and C_i is equal to P_i , character for character; or
- 2555 • P_i is a CAGECodeOrDODAAC, and C_i is equal to P_i ; or
- 2556 • P_i is a RangeComponent [$lo-hi$], and $lo \leq C_i \leq hi$; or
- 2557 • P_i is a StarComponent (and C_i is anything at all)

2558 Then the EPC Tag URI is a member of the set denoted by the EPC Pattern URI if and only if
2559 C_i matches P_i for all $1 \leq i \leq n$.

2560 The set of pure identity EPCs denoted by a specific EPC Pure Identity URI is defined by a
2561 similar decision procedure, which says whether a given EPC Pure Identity URI belongs to
2562 the set denoted by the EPC Pure Identity Pattern URI.

2563 Let $urn:epc:idpat:SchemeName:P1.P2...Pn$ be an EPC Pure Identity Pattern
2564 URI. Let $urn:epc:id:SchemeName:C1.C2...Cn$ be an EPC Pure Identity URI,
2565 where the *SchemeName* field of both URIs is the same. The number of components (n)
2566 depends on the value of *SchemeName*.

2567 Then the EPC Pure Identity URI is a member of the set denoted by the EPC Pure Identity
2568 Pattern URI if and only if C_i matches P_i for all $1 \leq i \leq n$, where “matches” is as defined
2569 above.

2570 **6 Background Information (non-normative)**

2571 This document draws from the previous work at the Auto-ID Center, and we recognize the
2572 contribution of the following individuals: David Brock (MIT), Joe Foley (MIT), Sunny Siu
2573 (MIT), Sanjay Sarma (MIT), and Dan Engels (MIT). In addition, we recognize the
2574 contribution from Steve Rehling (P&G) on EPC to GTIN mapping.

2575 The following papers capture the contributions of these individuals:

- 2576 • Engels, D., Foley, J., Waldrop, J., Sarma, S. and Brock, D., "The Networked Physical
2577 World: An Automated Identification Architecture"
2578 2nd IEEE Workshop on Internet Applications (WIAPP '01),
2579 (<http://csdl.computer.org/comp/proceedings/wiapp/2001/1137/00/11370076.pdf>)
- 2580 • Brock, David. "The Electronic Product Code (EPC), A Naming Scheme for Physical
2581 Objects", 2001. (<http://www.autoidlabs.org/whitepapers/MIT-AUTOID-WH-002.pdf>)
- 2582 • Brock, David. "The Compact Electronic Product Code; A 64-bit Representation of the
2583 Electronic Product Code", 2001. (<http://www.autoidlabs.com/whitepapers/MIT-AUTOID-WH-008.pdf>)
2584
- 2585 • D. Engels, “The Use of the Electronic Product Code™,” MIT Auto-ID Center Technical
2586 Report MIT-TR007, February 2003, (<http://www.autoidlabs.com/whitepapers/mit-autoid-tr009.pdf>)
2587

2588 • R. Moats, “URN Syntax,” Internet Engineering Task Force Request for Comments RFC-
2589 2141, May 1997, (<http://www.ietf.org/rfc/rfc2141.txt>)

2590 **7 References**

2591 [EAN.UCCGS] “General EAN.UCC Specifications.” Version 6.0, EAN.UCC, IncTM.

2592 [MIT-TR009] D. Engels, “The Use of the Electronic Product CodeTM,” MIT Auto-ID Center
2593 Technical Report MIT-TR007, February 2003, [http://www.autoidlabs.com/whitepapers/mit-](http://www.autoidlabs.com/whitepapers/mit-autoid-tr009.pdf)
2594 [autoid-tr009.pdf](http://www.autoidlabs.com/whitepapers/mit-autoid-tr009.pdf)

2595 [RFC2141] R. Moats, “URN Syntax,” Internet Engineering Task Force Request for
2596 Comments RFC-2141, May 1997, <http://www.ietf.org/rfc/rfc2141.txt>.

2597 [DOD Constructs] “United States Department of Defense Suppliers’ Passive RFID
2598 Information Guide,” <http://www.dodrfid.org/suppliernguide.htm>

2599 [Gen2 Specification] “EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF
2600 RFID Protocol for Communications at 860 MHz-960MHz Version 1.0.9”

2601

2602 **8 Appendix A: Encoding Scheme Summary Tables (non-**
 2603 **normative)**

2604

SGTIN Summary						
SGTIN-96	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
	8	3	3	20-40	24 - 4	38
	0011 0000 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	9,999,999 – 9 (Max .decimal range**)	274,877,906,943 (Max .decimal value)
SGTIN-198	Header	Filter Value	Partition	Company Prefix	Item Reference	Serial Number
	8	3	3	20-40	24 - 4	140
	0011 0110 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	9,999,999 – 9 (Max .decimal range**)	Up to 20 alphanumeric characters
Filter Values (Non-normative)		SGTIN Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Indicator Digit and Item Reference	
All Others	000		Bits	Digits	Bits	Digit
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

2605
2606

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

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

2608
2609

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

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

SGLN Summary						
SGLN-96	Header	Filter Value	Partition	Company Prefix	Location Reference	Extension Component
	8	3	3	20-40	21-1	41
	0011 0010 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	999,999 – 0 (Max. decimal range**)	2,199,023,255,551 (Max Decimal Value) Recommend: Min=1 Max=999,999,999,999 Reserved=0 All bits shall be set to 0 when an Extension Component is not encoded signifying GLN only.
SGLN-195	Header	Filter Value	Partition	Company Prefix	Location Reference	Extension component
	8	3	3	20-40	21-1	140
	0011 1001 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	999,999 – 0 (Max. decimal range**)	Up to 20 alphanumeric characters If Extension Component is not used these 140 bits shall all be set to binary 0
Filter Values (Non-normative)		SGLN Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Location Reference	
			Bits	Digits	Bits	Digit
All Others	000					
Physical Location	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

2611 *Range of Location Reference field varies with the length of the Company Prefix

2612 **Range of Company Prefix and Location Reference fields vary according to contents of the Partition field.

GRAI Summary						
GRAI-96	Header	Filter Value	Partition	Company Prefix	Asset Type	Serial Number
	8	3	3	20-40	24 – 4	38
	0011 0011 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	999,999 – 0 (Max. decimal range**)	274,877,906,943 (Max. decimal value)
GRAI-170	Header	Filter Value	Partition	Company Prefix	Asset Type	Serial Number
	8	3	3	20-40	24 – 4	112
	0011 0111 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	999,999 – 0 (Max. decimal range**)	Up to 16 alphanumeric characters
Filter Values (Non-normative)		GRAI Partition Table				
Type	Binary Value	Partition Value	Company Prefix		Asset Type***	
			Bits	Digits	Bits	Digit
All Others	000					
Reserved	001	0	40	12	4	0
Reserved	010	1	37	11	7	1
Reserved	011	2	34	10	10	2
Reserved	100	3	30	9	14	3
Reserved	101	4	27	8	17	4
Reserved	110	5	24	7	20	5
Reserved	111	6	20	6	24	6

2614 *Range of Asset Type field varies with Company Prefix.

2615 **Range of Company Prefix and Asset Type fields vary according to contents of the Partition field.

2616 *** Explanation (non-normative): The Asset Type field of the GRAI-96 has four more bits than necessary given
 2617 the capacity of that field.

GIAI Summary					
GIAI-96	Header	Filter Value	Partition	Company Prefix	Individual Asset Reference
	8	3	3	20-40	62-42
	0011 0100 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range*)	4,611,686,018,427,387,903 - 4,398,046,511,103 (Max. decimal range*)
GIAI-202	Header	Filter Value	Partition	Company Prefix	Individual Asset Reference
	8	3	3	20-40	168-126
	0011 1000 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range*)	Up to 24 alphanumeric characters
Filter Values (To be confirmed)		GIAI Partition Table			
Type	Binary Value	Partition Value	Company Prefix		Individual Asset Reference
			Bits	Digits	Bits Digits
All Others	000				
Reserved	001	<GIAI-96>			
Reserved	010	0	40	12	42 12
Reserved	011	1	37	11	45 13
Reserved	100	2	34	10	48 14
Reserved	101	3	30	9	52 15
Reserved	110	4	27	8	55 16
Reserved	111	5	24	7	58 17
		6	20	6	62 18
		<GIAI-202>			
		0	40	12	148 18
		1	37	11	151 19
		2	34	10	154 20
		3	30	9	158 21
		4	27	8	161 22
		5	24	7	164 23
		6	20	6	168 24

2619 *Range of Company Prefix and Individual Asset Reference fields vary according to contents of the Partition field.

2620
2621
2622

9 Appendix B: TDS 1.3 EAN.UCC Identities Bit Allocation and Required Physical Tag Bit Length for Encoding (non-normative)

Memory Bank Names	Reserved Memory Bank	EPC Memory Bank					TID Memory Bank	User Memory Bank			
EPC Memory Bank		CRC-16	Protocol Control Bits			EPC Bits					
Protocol Control Bits			Length bits	RFU	Numbering Systems Identifier						
Bit Field	Reserved Memory bits	CRC-16 bits	Length bits	RFU bits	EPC/ISO Toggle bit	Reserved / AFI bits	EPC Header + Filter value bits + Partition value bits + Domain Identifier bits	Word Boundary Filler bits	TID bits	User Memory bits	Total bits required
EPC Identity Names											
GID-96	64	16	5	2	1	8	96	0	32	0	224
SGTIN-96	64	16	5	2	1	8	96	0	32	0	224
SGTIN-198	64	16	5	2	1	8	198	10	32	0	336
SSCC-96	64	16	5	2	1	8	96	0	32	0	224
SGLN-96	64	16	5	2	1	8	96	0	32	0	224
SGLN-195	64	16	5	2	1	8	195	13	32	0	336
GRAI-96	64	16	5	2	1	8	96	0	32	0	224
GRAI-170	64	16	5	2	1	8	170	6	32	0	304
GIAI-96	64	16	5	2	1	8	96	0	32	0	224
GIAI-202	64	16	5	2	1	8	202	6	32	0	336

2623
2624

2625 Notes:

2626 GIAI-202 may have shorter Domain Identifier bits (Company Prefix and Individual Asset
2627 Reference) which will shorten the total bit requirement to 302 bits.

2628 All the bits except for CRC-16 in the EPC Memory Bank requires encoding by application or
2629 process

2630 This table illustrates the total number of bits required in the three logical memories (TID,
2631 Reserved and EPC) to support the EAN,UCC identities listed. User memory is set to zero
2632 required bits to load a single identity in the tag. As larger memories are defined and the User
2633 memory method of allocation is defined in this standard, additional bits can be assigned to
2634 user memory.

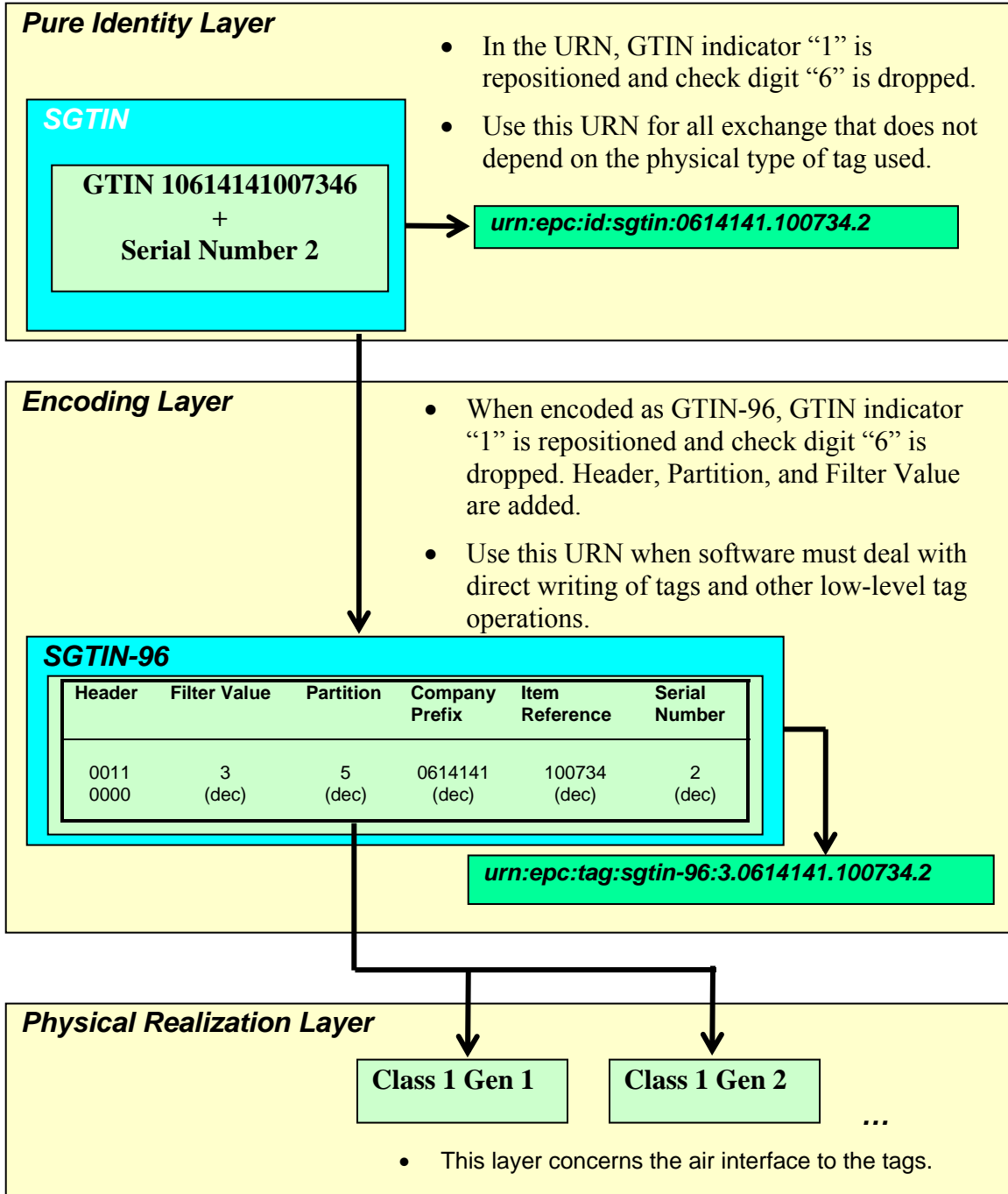
2635 The EPC bits includes the extra bits required to round up to a fill the last 16 bit word.

2636 The four identities; SGTIN-198, SGLN-195, GRAI-170 and GIAI-202 have been included in
2637 this standard to indicate to hardware vendors the user requirements for tag sizes and memory
2638 allocation required to support these longer identities. Please note that all three required more
2639 than 256 bits to contain all the fields required.

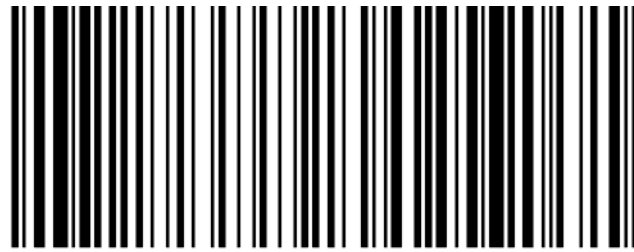
2640 The Generation two protocol allows for reserved commands that are anticipated to provide
2641 dynamic assignment of memory as well as fixed static memory assignment.

2642 **10 Appendix C: Example of a Specific Trade Item <SGTIN>**
 2643 **(non-normative)**

2644 This section presents an example of a specific trade item using SGTIN (Serialized GTIN).
 2645 Each representation serves a distinct purpose in the software stack. Generally, the highest
 2646 applicable level should be used. The GTIN used in the example is 10614141007346.



2647



2648

2649

2650

2651

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

2652

2653

2654

2655

2656

2657

2658

2659

2660

2661

2662

2663

2664

Explanation of SGTIN Filter Values (non-normative).

2665

2666

2667

2668

2669

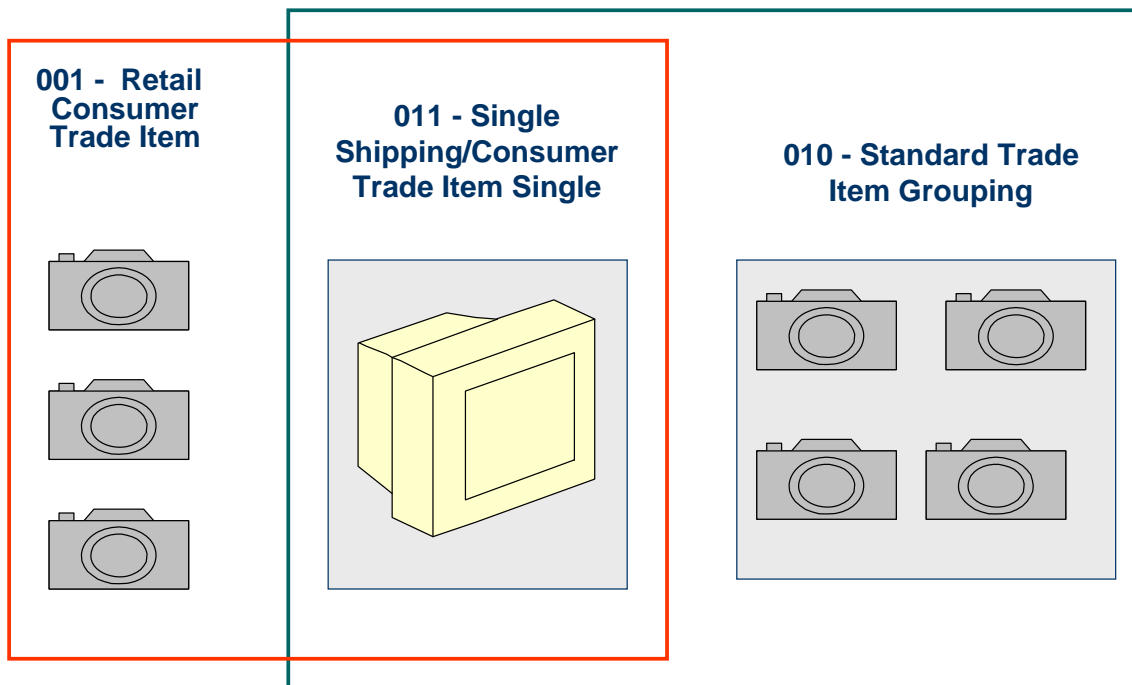
SGTINs can be assigned at several levels, including: item, inner pack, case, and pallet. RFID can read through cardboard, and reading un-needed tags can slow us down, so Filter Values are used to “filter in” desired tags, or “filter out” unwanted tags. Filter values are used within the key type (i.e.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

- (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.
- Header for SGTIN-96 is 00110000.
- Filter Value of 3 (Single Shipping/ Consumer Trade Item) was chosen for this example.
- 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.
- Indicator digit 1 is repositioned as the first digit in the Item Reference.
- Check digit 6 is dropped.

2670 now until the community gains more practical experience in their use. Therefore the three
2671 major categories of SGTIN filter values can be thought of in the following high level terms:

- 2672 • Single Unit: A Retail Consumer Trade Item
- 2673 • Not-a-single unit: A Standard Trade Item Grouping
- 2674 • Items that could be included in both categories: For example, a Single Shipping
2675 container that contains a Single Consumer Trade Item

Three Filter Values



2676

2677

2678

2679

2680

2681

11 Appendix D: Decimal values of powers of 2 Table (non-normative)

n	(2^n) ₁₀	n	(2^n) ₁₀
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	288,435,456	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		

2682

2683
2684

12 Appendix E: List of Abbreviations

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

2685
2686

2687 **13 Appendix F: General EAN.UCC Specifications (non-**
2688 **normative)**

2689 (Section 3 Definition of Element Strings and Section 3.7 EPCglobal Tag Data Standard.)

2690 This section provides GS1 approval of this version of the EPCglobal® Tag Data Standard
2691 with the following EAN.UCC Application Identifier definition restrictions:

2692 Companies should use the EAN.UCC specifications to define the applicable fields in
2693 databases and other ICT-systems.

2694 For EAN.UCC use of EPC96-bit tags, the following applies:

- 2695 • **AI (00) SSCC** (no restrictions)
- 2696 • **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is
2697 restricted to permit assignment of 274,877,906,943 numeric-only serial numbers)
- 2698 • **AI (414) GLN + AI (254) GLN Extension Component:** The Tag Data Standard V1.1 R1.27
2699 is approved for the use of GLN Extension with the restrictions specified in Section 2.4.6.1 of
2700 the General EAN.UCC Specifications..
- 2701 • **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier
2702 definition is restricted to permit assignment of 274,877,906,943 numeric-only serial numbers
2703 and the serial number element is mandatory.
- 2704 • **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier
2705 definition is restricted to permit assignment of 4,611,686,018,427,387,904 numeric-only serial
2706 numbers.

2707 For EAN.UCC use of EPC longer then 96-bit tags, the following applies:

- 2708 • **AI (00) SSCC** (no restrictions)
- 2709 • **AI (01) GTIN + AI (21) Serial Number:** (no restrictions)
- 2710 • **AI (414) GLN + AI (254) Extension Component:** (no restrictions).
- 2711 • **AI (8003) GRAI Serial Number:** (no restrictions)
- 2712 • **AI (8004) GIAI Serial Number:** (no restrictions)

15 Appendix G: EAN.UCC Alphanumeric Character Set (Normative)

ISO/IEC 646 Subset

Unique Graphic Character Allocations					
Graphic Symbol	Name	Hex Coded Representation	Graphic Symbol	Name	Hex Coded Representation
!	Exclamation mark	21	M	Capital letter M	4D
"	Quotation mark	22	N	Capital letter N	4E
%	Percent sign	25	O	Capital letter O	4F
&	Ampersand	26	P	Capital letter P	50
'	Apostrophe	27	Q	Capital letter Q	51
(Left parenthesis	28	R	Capital letter R	52
)	Right parenthesis	29	S	Capital letter S	53
*	Asterisk	2A	T	Capital letter T	54
+	Plus sign	2B	U	Capital letter U	55
,	Comma	2C	V	Capital letter V	56
-	Hyphen/Minus	2D	W	Capital letter W	57
.	Full stop	2E	X	Capital letter X	58
/	Solidus	2F	Y	Capital letter Y	59
0	Digit zero	30	Z	Capital letter Z	5A
1	Digit one	31	_	Low line	5F
2	Digit two	32	a	Small letter a	61
3	Digit three	33	b	Small letter b	62
4	Digit four	34	c	Small letter c	63
5	Digit five	35	d	Small letter d	64
6	Digit six	36	e	Small letter e	65
7	Digit seven	37	f	Small letter f	66
8	Digit eight	38	g	Small letter g	67
9	Digit nine	39	h	Small letter h	68
:	Colon	3A	i	Small letter i	69
;	Semicolon	3B	j	Small letter j	6A
<	Less-than sign	3C	k	Small letter k	6B
=	Equals sign	3D	l	Small letter l	6C
>	Greater-than sign	3E	m	Small letter m	6D
?	Question mark	3F	n	Small letter n	6E
A	Capital letter A	41	o	Small letter o	6F
B	Capital letter B	42	p	Small letter p	70

C	Capital letter C	43	q	Small letter q	71
D	Capital letter D	44	r	Small letter r	72
E	Capital letter E	45	s	Small letter s	73
F	Capital letter F	46	t	Small letter t	74
G	Capital letter G	47	u	Small letter u	75
H	Capital letter H	48	v	Small letter v	76
I	Capital letter I	49	w	Small letter w	77
J	Capital letter J	4A	x	Small letter x	78
K	Capital letter K	4B	y	Small letter y	79
L	Capital letter L	4C	z	Small letter z	7A

2715 Notes

2716 Readers should be aware that this table is derived from [EAN.UCCGS] and may include
 2717 discrepancy with the original specification at any given time. Readers are advised to always
 2718 consult the original specification upon implementation.

2719 This table specifies the allowed subset of ISO/IEC 646 characters that shall be used for
 2720 encoding alphanumeric Serial Number/Extension Component in this standard. The SGTIN-
 2721 198, SGLN-195, GRAI-170 and GIAI-202 encodings use this table.

2722 Each entry in this table gives a 7-bit code for a character, expressed in hexadecimal. For
 2723 example, “Capital Letter K” has a 7-bit code of 1001011, expressed as “4B” in the table.

2724 The 7-bit codes in this table are identical to ISO/IEC 646 (ASCII) character codes.