



1

2

## EPCglobal Tag Data Standards Version 1.4

3

Ratified on June 11, 2008

4

### Disclaimer

6

EPCglobal Inc<sup>TM</sup> is providing this document as a service to interested industries. This document was developed through a consensus process of interested parties. Although efforts have been to assure that the document is correct, reliable, and technically accurate, EPCglobal Inc. makes NO WARRANTY, EXPRESS OR IMPLIED, THAT THIS DOCUMENT IS CORRECT, WILL NOT REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE, OR WILL BE SUITABLE FOR ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR OTHERWISE. Use of this document is with the understanding that EPCglobal Inc. has no liability for any claim to the contrary, or for any damage or loss of any kind or nature.

16

17

### Copyright notice

18

© 2006, 2007, 2008 EPCglobal Inc.

19

All rights reserved. Unauthorized reproduction, modification, and/or use of this document is not permitted. Requests for permission to reproduce should be addressed to [epcglobal@epcglobalinc.org](mailto:epcglobal@epcglobalinc.org).

20

21

22

23

EPCglobal Inc.<sup>TM</sup> is providing this document as a service to interested industries. This document was developed through a consensus process of interested parties. Although efforts have been to assure that the document is correct, reliable, and technically accurate, EPCglobal Inc. makes NO WARRANTY, EXPRESS OR IMPLIED, THAT THIS DOCUMENT IS CORRECT, WILL NOT REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE, OR WILL BE SUITABLE FOR ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR OTHERWISE. Use of this Document is with the understanding that EPCglobal Inc. has no liability for any claim to the contrary, or for any damage or loss of any kind or nature

24

25

26

27

28

29

30

31

### DOCUMENT HISTORY

32

Document Number:	
------------------	--

<b>Document Version:</b>	1.4
<b>Document Date :</b>	June 11, 2008

33  
34  
35  
36  
37

### Document Summary

<b>Document Title:</b>	<b>EPC™ Tag Data Standards Version 1.4</b>
<b>Owner:</b>	Tag Data and Translation Standard Work Group
<b>Status:</b>	( <i>check one box</i> )      DRAFT      X Approved

38  
39  
40

### Document Change History

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
10/19/2007	1.4	New Identities	• GSRN-96, GDTI-96, GDTI-113
03/28/2008	1.4	Recognition	• Appendix G: Participants and Opted-in Companies
04/07/08	1.4	Approval	• Proposed Spec. advanced to RS by TSC & BSC
			•

41

## 42 **Abstract**

43 This document defines the EPC Tag Data Standards version 1.4. It applies to RFID tags  
44 conforming to “EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID  
45 Protocol for Communications at 860 MHz-960MHz Version 1.1.0” (“Gen2 Specification”).  
46 Such tags will be referred to as “Gen 2 Tags” in the remainder of this document. These  
47 standards define completely that portion of EPC tag data that is standardized, including how  
48 that data is encoded on the EPC tag itself (i.e. the EPC Tag Encodings), as well as how it is  
49 encoded for use in the information systems layers of the EPC Systems Network (i.e. the EPC  
50 URI or Uniform Resource Identifier Encodings).

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

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

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

68

## 69 **Status of this document**

70 This section describes the status of this document at the time of its publication. Other  
71 documents may supersede this document. The latest status of this document series is  
72 maintained at EPCglobal. See <http://www.epcglobalinc.org/standards/tds/>.

73

74 On June 11<sup>th</sup>, the Recommended Specification was reviewed and ratified by the EPCglobal  
75 Board. It is now a Ratified Standard and can be fully implemented and referenced.

76

77 Further comments or potential errata found pertaining to this document should be sent to the  
78 EPCglobal Software Action Group’s Tag Data & Translation Standards Working Group at  
79 the following mailing list: [sag\\_tdts\\_wg@lists.epcglobalinc.org](mailto:sag_tdts_wg@lists.epcglobalinc.org) .

80

## 81 **Changes from Previous Versions**

### 82 Version 1.4

83 This update to the Tag Data Standards provides support for two GS1 identities: The Global  
84 Service Relation Number (GSRN) and the Global Document Type Identifier (GDTI).

85 Changes are as follows

86

- 87 1. Sections 2.1.2.6 (GSRN) and 2.1.2.7 (GDTI) describe the new Identity Types.
- 88 2. The Header Table 1 has three new entries: x2C for GSRN-96, x2D for GDTI-96 and  
89 x3A for GDTI-113.
- 90 3. A new filter value of “100” has been added to the SGTIN filter Table in Section  
91 3.5.1.
- 92 4. The Encoding and Decoding procedures for the GRAI-96 and GRAI-170 have been  
93 changed in sections 3.8.1.1, 3.8.1.2, 3.8.2.1 and 3.8.2.2 to eliminate a leading 0 that  
94 has been encoded as a filler 0 character in the GRAI. This leading “0” is not part of  
95 the GRAI in the GS1 General Specifications.
- 96 5. Changes from a value “168-126” to a value of “168-148” in table 22 in section 3.9.2  
97 and the GIAI Summary table in Appendix A have been made to correct an error in  
98 version 1.3.1.
- 99 6. Section 3.10 is the detailed definition and encoding and decoding procedures for the  
100 Global Service Relation Number (GSRN).
- 101 7. Section 3.11 is the detailed definition and encoding and decoding procedures for the  
102 Global Document Type Identifier (GDTI).
- 103 8. Section 4 has been updated to include the URI forms for the new identities.
- 104 9. Section 5 has been updated to include the translations for the new identities
- 105 10. Appendix A and B have been updated with new tables for the new identities
- 106 11. References to EAN.UCC have been changed to GS1 throughout the document.
- 107 12. The Sunset date of July 1, 2009 for 64 bit Header values has been added.
- 108 13. Deleted Appendix B which was the bit allocation table because it is too  
109 implementation specific for this TDTS document. This table, if needed, will be  
110 published by EPCglobal.

111

### 112 Version 1.3.1

113

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

116

- 117 1. In section 3.8.2.2 GRAI-170 Decoding Procedure, the bit numbering has been  
118 corrected. For instance “00110111  $b_{162}b_{161} \dots b_0$ “ has been corrected to read  
119 “00110111  $b_{161}b_{160} \dots b_0$ “ and so forth throughout the section.
- 120 2. The GIAI-202 Table 23 and the Associated Summary Table in Appendix A did not  
121 add up to a total of 188 bits for each Company Prefix/Individual Asset Reference  
122 which is what the encoding/decoding procedure expects. The Individual Asset  
123 Reference Bits column has been changed so each row adds to 188 bits. For example,  
124 for Partition value 0 the Individual Asset Reference bits value “126” was changed to  
125 “148”.
- 126 3. An addition error in the Appendix B table, SGLN-195 row, has been corrected. The  
127 Total bits required column was changed from 333 to 336.
- 128 4. A typographical error in line three of the section 3.8.1.1 GRAI-96 Encoding  
129 Procedure has been corrected. The formula “ $15 \leq K \leq 0$ ” was replaced with  
130 “ $15 \leq K \leq 30$ ”.
- 131 5. In Section 5.4 (Gen 2 Tag EPC Memory into Tag or Raw URI) step 8 line 4 a  
132 missing dot (.) character after the value of A has been corrected.
- 133 6. The arrows in Appendix B between the Bar Code symbol and the SGTIN-96 have  
134 been adjusted to reflect the connections between the Company Prefix, Item Reference  
135 and Serial Number.

136

### 137 Version 1.3

138

139 This Tag Data Standards Version 1.3 is aimed for use in Gen 2 Tags, whereas the previous  
140 Version 1.1, was aimed for use in UHF Class 1 Generation 1 tags. Version 1.3 maintains  
141 compatibility with version 1.1 in the identity level. In other words, this version will continue  
142 to support the GS1 system and DoD identity types.

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

- 144 1. The deprecation of 64 bit encodings.
- 145 2. The elimination of tiered header rules.
- 146 3. The encoding of EPC to fit the structure of Gen 2 Tags
- 147 4. The addition of the Extension Component to the SGLN
- 148 5. Addition of SGTIN-198, SGLN-195, GRAI-170, GIAI-202 and corresponding  
149 changes in URI expression for alpha-numeric serial number encoding.

150

151 **Table of Contents**

152 1 Introduction ..... 9

153 2 Identity Concepts..... 10

154 2.1 Pure Identities ..... 12

155 2.1.1 General Types ..... 12

156 2.1.2 GS1 System Identity Types ..... 13

157 2.1.2.1 Serialized Global Trade Item Number (SGTIN)..... 13

158 2.1.2.2 Serial Shipping Container Code (SSCC) ..... 15

159 2.1.2.3 Serialized Global Location Number (SGLN)..... 16

160 2.1.2.4 Global Returnable Asset Identifier (GRAI) ..... 18

161 2.1.2.5 Global Individual Asset Identifier (GIAI)..... 18

162 2.1.2.6 Global Service Relation Number (GSRN)..... 19

163 2.1.2.7 Global Document Type Identifier (GDTI)..... 20

164 2.1.3 DoD Identity Type ..... 20

165 3 EPC Tag Bit-level Encodings ..... 20

166 3.1 Headers ..... 21

167 3.2 Use of EPCs on UHF Class 1 Generation 2 Tags..... 23

168 3.2.1 EPC Memory Contents ..... 24

169 3.2.2 The Length Bits..... 25

170 3.3 Notational Conventions ..... 26

171 3.4 General Identifier (GID-96)..... 27

172 3.4.1.1 GID-96 Encoding Procedure ..... 28

173 3.4.1.2 GID-96 Decoding Procedure..... 28

174 3.5 Serialized Global Trade Item Number (SGTIN)..... 29

175 3.5.1 SGTIN-96 ..... 29

176 3.5.1.1 SGTIN-96 Encoding Procedure ..... 31

177 3.5.1.2 SGTIN-96 Decoding Procedure ..... 32

178 3.5.2 SGTIN-198 ..... 33

179 3.5.2.1 SGTIN-198 Encoding Procedure ..... 34

180 3.5.2.2 SGTIN-198 Decoding Procedure ..... 35

181 3.6 Serial Shipping Container Code (SSCC)..... 36

182	3.6.1	SSCC-96 .....	36
183	3.6.1.1	SSCC-96 Encoding Procedure .....	38
184	3.6.1.2	SSCC-96 Decoding Procedure .....	38
185	3.7	Serialized Global Location Number (SGLN).....	39
186	3.7.1	SGLN-96.....	40
187	3.7.1.1	SGLN-96 Encoding Procedure.....	42
188	3.7.1.2	SGLN-96 Decoding Procedure .....	43
189	3.7.2	SGLN-195.....	44
190	3.7.2.1	SGLN-195 Encoding Procedure.....	45
191	3.7.2.2	SGLN-195 Decoding Procedure .....	45
192	3.8	Global Returnable Asset Identifier (GRAI).....	46
193	3.8.1	GRAI-96 .....	47
194	3.8.1.1	GRAI-96 Encoding Procedure .....	49
195	3.8.1.2	GRAI-96 Decoding Procedure .....	49
196	3.8.2	GRAI-170 .....	50
197	3.8.2.1	GRAI-170 Encoding Procedure .....	51
198	3.8.2.2	GRAI-170 Decoding Procedure.....	52
199	3.9	Global Individual Asset Identifier (GIAI).....	53
200	3.9.1	GIAI-96.....	53
201	3.9.1.1	GIAI-96 Encoding Procedure.....	55
202	3.9.1.2	GIAI-96 Decoding Procedure .....	56
203	3.9.2	GIAI-202.....	56
204	3.9.2.1	GIAI-202 Encoding Procedure.....	58
205	3.9.2.2	GIAI-202 Decoding Procedure .....	59
206	3.10	Global Service Relation Number (GSRN).....	60
207	3.10.1	GSRN-96 .....	60
208	3.10.1.1	GSRN-96 Encoding Procedure .....	62
209	3.10.1.2	GSRN-96 Decoding Procedure.....	62
210	3.11	Global Document Type Identifier (GDTI).....	63
211	3.11.1	GDTI-96 .....	63
212	3.11.1.1	GDTI-96 Encoding Procedure .....	65
213	3.11.1.2	GDTI-96 Decoding Procedure.....	66

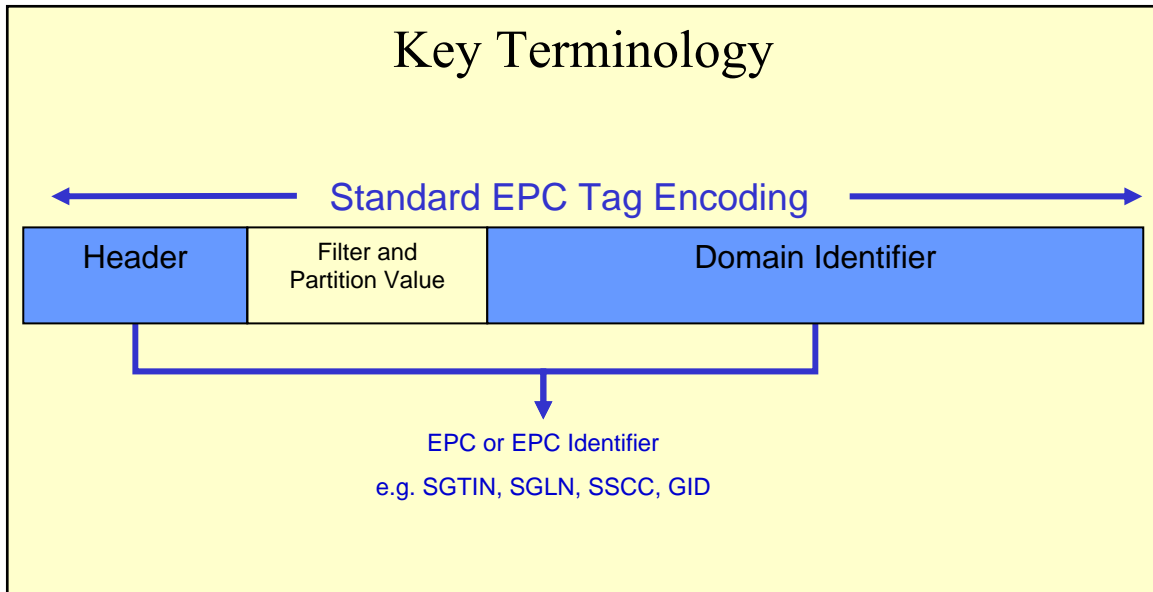
214	3.11.2	GDTI-113 .....	66
215	3.11.2.1	GDTI-113 Encoding Procedure .....	67
216	3.11.2.2	GDTI-113 Decoding Procedure .....	68
217	3.12	DoD Tag Data Constructs .....	69
218	3.12.1	DoD-96 .....	69
219	4	URI Representation .....	70
220	4.1	URI Forms for Pure Identities .....	71
221	4.2	URI Forms for Related Data Types .....	73
222	4.2.1	URIs for EPC Tags .....	73
223	4.2.2	URIs for Raw Bit Strings Arising From Invalid Tags .....	74
224	4.2.2.1	Use of the Raw URI with Gen 2 Tags .....	75
225	4.2.2.2	The Length Field of a Raw URI when using Gen 2 Tags (non-normative) .....	76
226	4.2.3	URIs for EPC Patterns .....	76
227	4.2.4	URIs for EPC Pure Identity Patterns .....	77
228	4.3	Syntax .....	78
229	4.3.1	Common Grammar Elements .....	78
230	4.3.2	EPCGID-URI .....	79
231	4.3.3	SGTIN-URI .....	79
232	4.3.4	SSCC-URI .....	79
233	4.3.5	SGLN-URI .....	79
234	4.3.6	GRAI-URI .....	79
235	4.3.7	GIAI-URI .....	80
236	4.3.8	GSRN-URI .....	80
237	4.3.9	GDTI-URI .....	80
238	4.3.10	EPC Tag URI .....	81
239	4.3.11	Raw Tag URI .....	82
240	4.3.12	EPC Pattern URI .....	82
241	4.3.13	EPC Identity Pattern URI .....	83
242	4.3.14	DoD Construct URI .....	84
243	4.3.15	Summary (non-normative) .....	85
244	5	Translation between EPC-URI and Other EPC Representations .....	88
245	5.1	Bit string into EPC-URI (pure identity) .....	89

246	5.2	Bit String into Tag or Raw URI.....	91
247	5.3	Gen 2 Tag EPC Memory into EPC-URI (pure identity) .....	94
248	5.4	Gen 2 Tag EPC Memory into Tag or Raw URI .....	95
249	5.5	URI into Bit String .....	95
250	5.6	URI into Gen 2 Tag EPC Memory .....	100
251	6	Semantics of EPC Pattern URIs .....	100
252	7	Background Information (non-normative).....	101
253	8	References .....	102
254		Appendix A: Encoding Scheme Summary Tables (non-normative) .....	103
255		Appendix B: Example of a Specific Trade Item <SGTIN> (non-normative) .....	110
256		Appendix C: Decimal values of powers of 2 Table (non-normative).....	113
257		Appendix D: List of Abbreviations.....	114
258		Appendix E: GS1 General Specifications Version 7.1 (non-normative).....	115
259		Appendix F: GS1 Alphanumeric Character Set.....	116
260		Appendix G: Acknowledgement of Contributors and Companies Opted-in during the	
261		Creation of this Standard (Informative).....	117

## 262 **1 Introduction**

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

268 The EPC Identifier is a meta-coding scheme designed to support the needs of various  
269 industries by accommodating both existing coding schemes where possible and defining\*  
270 new schemes where necessary. The various coding schemes are referred to as Domain  
271 Identifiers, to indicate that they provide object identification within certain domains such as  
272 a particular industry or group of industries. As such, the Electronic Product Code represents  
273 a family of coding schemes (or “namespaces”) and a means to make them unique across all  
274 possible EPC-compliant tags. The various GS1 coding schemes and their associated data  
275 structures and applications are defined in the Section 8 reference [GS1GS]. These concepts  
276 are depicted in the chart below.



277

278

**Figure A. EPC Terminology**

279

280 In this version of the EPCglobal Tag Data Standard 1.4 – the specific coding schemes  
 281 include a General Identifier (GID), a serialized version of the Global Trade Item Number  
 282 (GTIN®), the Serial Shipping Container Code (SSCC®), the Global Location Number  
 283 (GLN®), the Global Returnable Asset Identifier (GRAI®), the Global Individual Asset  
 284 Identifier (GIAI®), the Global Service Relation Number (GSRN®), the Global Document  
 285 Type Identifier (GDTI®) and the DOD Construct.

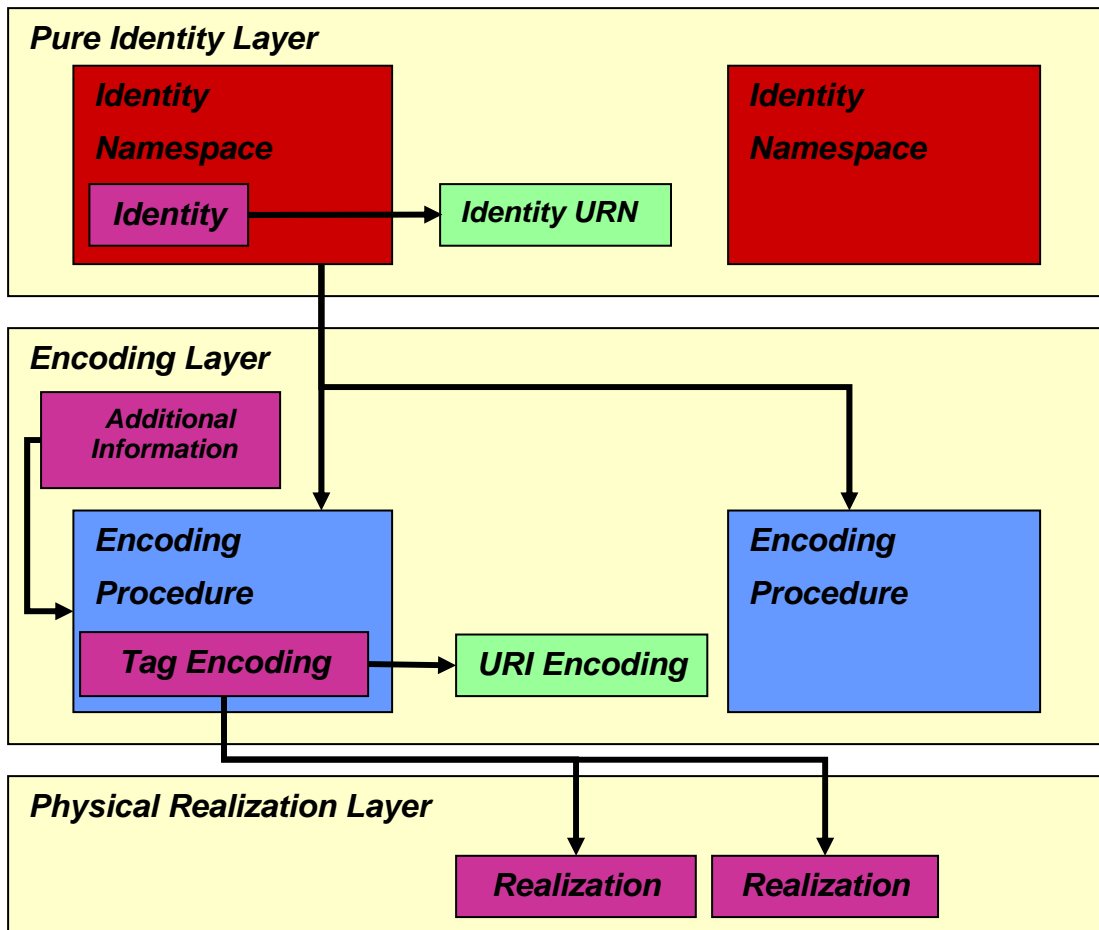
286 In the following sections, we will describe the structure and organization of the EPC and  
 287 provide illustrations to show its recommended use.

288 The EPCglobal Tag Data Standard 1.4 has been approved by GS1 with the restrictions  
 289 outlined in the GS1 General Specifications (Version 7.1) Section 3.7, which is excerpted into  
 290 Tag Data Standard Appendix E.

291 The latest version of this specification can be obtained from EPCglobal at  
 292 <http://www.epcglobalinc.org/standards/tds/>

## 293 **2 Identity Concepts**

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



298

299

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

300

- Pure identity -- the identity associated with a specific physical or logical entity, independent of any particular encoding vehicle such as an RF tag, bar code or database field. As such, a pure identity is an abstract name or number used to identify an entity. A pure identity consists of the information required to uniquely identify a specific entity, and no more.

301

302

303

304

305

- Identity URI -- a representation of a pure identity as a Uniform Resource Identifier (URI). A URI is a character string representation that is commonly used to exchange identity data between software components of a larger system.

306

307

308

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

309

310

311

312

313

314

315

- Physical Realization of an Encoding -- an encoding rendered in a concrete implementation suitable for a particular machine-readable form, such as a specific kind

316

317 of RF tag or specific database field. A given encoding may have a number of possible  
318 physical realizations.

319 For example, the Serial Shipping Container Code (SSCC) format as defined by the GS1  
320 System is an example of a pure identity. An SSCC encoded into the EPC-SSCC 96-bit  
321 format is an example of an encoding. That 96-bit encoding, written onto a UHF Class 1 RF  
322 Tag, is an example of a physical realization.

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

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

## 333 **2.1 Pure Identities**

334 This section defines the pure identity types for which this document specifies encoding  
335 schemes.

### 336 **2.1.1 General Types**

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

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

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

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

## 354 **2.1.2 GS1 System Identity Types**

355 This version of the EPC Tag Data Standards defines seven EPC identity types derived from  
356 the GS1 System family of product codes, each described in the subsections below.

357 The rules regarding the usage of the GS1 codes can be found in the GS1 General  
358 Specifications. This document only explains the incorporation of these numbers in EPC tags.

359 GS1 System codes have a common structure, consisting of a fixed number of decimal digits  
360 that encode the identity, plus one additional “check digit” which is computed algorithmically  
361 from the other digits. Within the non-check digits, there is an implicit division into two  
362 fields: a Company Prefix assigned by GS1 to a managing entity, and the remaining digits,  
363 which are assigned by the managing entity. (The digits apart from the Company Prefix are  
364 called by a different name by each of the GS1 System codes.) The number of decimal digits  
365 in the Company Prefix varies from 6 to 12 depending on the particular Company Prefix  
366 assigned. The number of remaining digits therefore varies inversely so that the total number  
367 of digits is fixed for a particular GS1 System code type.

368 The GS1 recommendations for the encoding of GS1 System identities into bar codes, as well  
369 as for their use within associated data processing software, stipulate that the digits  
370 comprising a GS1 System code should always be processed together as a unit, and not parsed  
371 into individual fields. This recommendation, however, is not appropriate within the EPC  
372 Network, as the ability to divide a code into the part assigned to the managing entity (the  
373 Company Prefix in GS1 System types) versus the part that is managed by the managing  
374 entity (the remainder) is essential to the proper functioning of the Object Name Service  
375 (ONS). In addition, the ability to distinguish the Company Prefix is believed to be useful in  
376 filtering or otherwise securing access to EPC-derived data. Hence, the EPC Tag Encodings  
377 for GS1 code types specified herein deviate from the aforementioned recommendations in  
378 the following ways:

- 379 • EPC Tag Encodings carry an explicit division between the Company Prefix and the  
380 remaining digits, with each individually encoded into binary. Hence, converting from  
381 the traditional decimal representation of a GS1 System code and an EPC Tag Encoding  
382 requires independent knowledge of the length of the Company Prefix.
- 383 • EPC Tag Encodings do not include the check digit. Hence, converting from an EPC Tag  
384 Encoding to a traditional decimal representation of a code requires that the check digit  
385 be recalculated from the other digits.

### 386 **2.1.2.1 Serialized Global Trade Item Number (SGTIN)**

387 The Serialized Global Trade Item Number is a new identity type based on the GS1 Global  
388 Trade Item Number (GTIN) code defined in the GS1 General Specifications. A GTIN by  
389 itself does not fit the definition of an EPC pure identity, because it does not uniquely identify  
390 a single physical object. Instead, a GTIN identifies a particular class of object, such as a  
391 particular kind of product or SKU.

392 *All representations of SGTIN support the full 14-digit GTIN format. This means that the zero*  
393 *indicator-digit and leading zero in the Company Prefix for GTIN-12, and the zero indicator-*  
394 *digit for GTIN-13, can be encoded and interpreted accurately from an EPC Tag Encoding.*

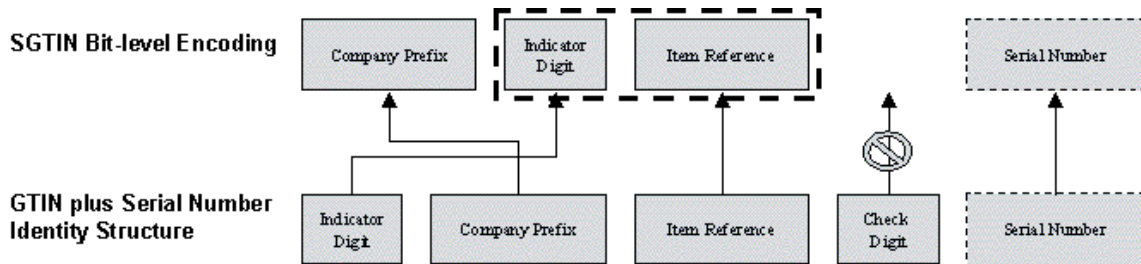
395 *GTIN-8 is not currently supported in EPC, but would be supported in full 14-digit GTIN*  
396 *format as well.*

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

401 The SGTIN consists of the following information elements:

- 402 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the  
403 same as the Company Prefix digits within a GS1 GTIN decimal code.
- 404 • The *Item Reference*, assigned by the managing entity to a particular object class. The  
405 Item Reference for the purposes of EPC Tag Encoding is derived from the GTIN by  
406 concatenating the Indicator Digit of the GTIN and the Item Reference digits, and  
407 treating the result as a single integer.
- 408 • The *Serial Number*, assigned by the managing entity to an individual object. The serial  
409 number is not part of the GTIN code, but is formally a part of the SGTIN.

410



411

412

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

415 The SGTIN is not explicitly defined in the GS1 General Specifications. However, it may be  
416 considered equivalent to a GS1-128 bar code that contains both a GTIN (Application  
417 Identifier 01) and a serial number (Application Identifier 21). GS1-128 and the term  
418 Application Identifier (AI) and the associated data structures and applications are defined in  
419 the Section 8 reference [GS1GS]. Serial numbers in AI 21 consist of one to twenty  
420 characters, where each character can be a digit, uppercase or lowercase letter, or one of a  
421 number of allowed punctuation characters. The complete set of characters allowed is  
422 illustrated in Appendix F. The complete AI 21 syntax is supported by the pure identity URI  
423 syntax specified in Section 4.3.1.

424 When representing serial numbers in 96-bit tags, however, only a subset of the serial  
425 numbers allowed in the GS1 General Specifications for Application Identifier 21 are  
426 permitted. Specifically, the permitted serial numbers are those consisting of one or more  
427 digits with no leading zeros, and whose value when considered as an integer fits within the  
428 range restrictions of the 96-bit tag encodings.

429 While these limitations exist for 96-bit tag encodings, other tag encodings allow a wider  
430 range of serial numbers. Therefore, application authors and database designers should take  
431 the GS1 specifications for Application Identifier 21 into account in order to accommodate  
432 the full range of allowed serial numbers.

433 For the requirement of using longer serial number, or alphabet and other non numeric  
434 codings allowed in Application Identifier 21, this specification includes a 198-bit tag  
435 encoding for SGTIN.

436 *Explanation (non-normative): The restrictions are necessary for 96-bit tags in order for*  
437 *serial numbers to fit within the small number of bits available in commonly available 96-bit*  
438 *tags. The serial number range is restricted to numeric values and alphanumeric serial*  
439 *numbers are disallowed. Leading zeros are forbidden so that the serial number can be*  
440 *considered as a decimal integer when encoding the integer value in binary. By considering*  
441 *it to be a decimal integer, "00034", "034", or "34" (for example) can't be distinguished as*  
442 *different integer values. In order to insure that every encoded value can be decoded*  
443 *uniquely, serial numbers can't have leading zeros. Then, when the bits*  
444 *00000000000000000000000010010 on the tag are seen, the serial number as "34" (not "034" or*  
445 *"00034") is decoded.*

#### 446 **2.1.2.2 Serial Shipping Container Code (SSCC)**

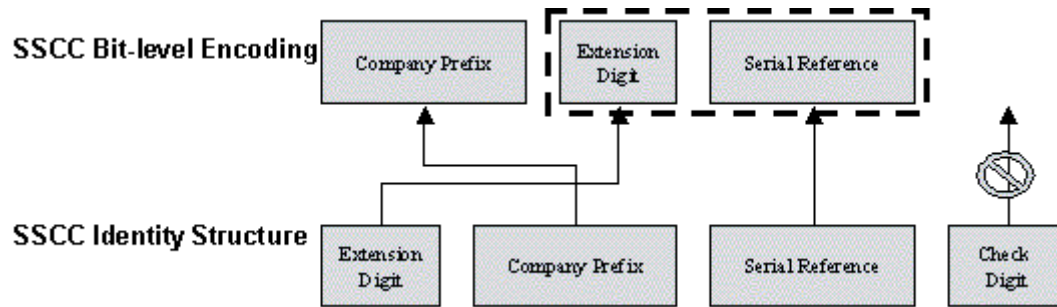
447 The Serial Shipping Container Code (SSCC) is defined by the GS1 General Specifications.  
448 Unlike the GTIN, the SSCC is already intended for assignment to individual objects and  
449 therefore does not require any additional fields to serve as an EPC pure identity.

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

456 The SSCC consists of the following information elements:

- 457 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the  
458 same as the Company Prefix digits within a GS1 SSCC decimal code.
- 459 • The *Serial Reference*, assigned uniquely by the managing entity to a specific shipping  
460 unit. The Serial Reference for the purposes of EPC Tag Encoding is derived from the  
461 SSCC by concatenating the Extension Digit of the SSCC and the Serial Reference  
462 digits, and treating the result as a single integer.

463



464

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

### 466 2.1.2.3 Serialized Global Location Number (SGLN)

467 The Global Location Number (GLN) is defined by the GS1 General Specifications as an  
 468 identifier of physical or legal entities.

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

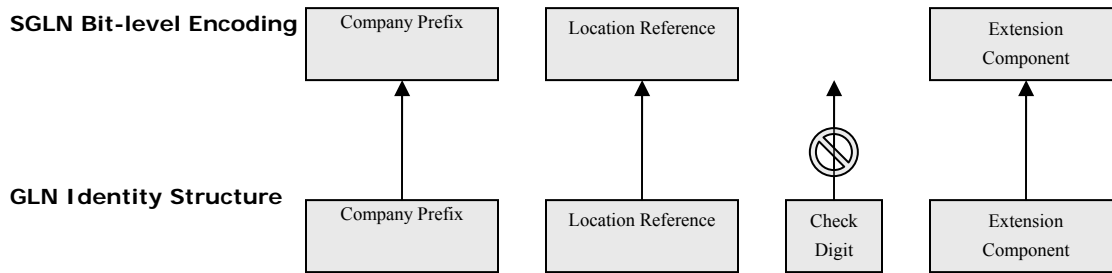
473 Within the GS1 system, high capacity data carriers use Application Identifiers (AI) to  
 474 distinguish data elements encoded within a single data carrier. The GLN can be associated  
 475 with many AI’s including physical location, ship to location, invoice to location etc.

476 Recognizing these variables, the EPC SGLN (serialized GLN) represents only the physical  
 477 location sub-type of GLN AI (414). The serial component is represented by the GLN  
 478 Extension AI (254). Rules regarding the allocation of a SGLN can be found within the GS1  
 479 General Specifications.

480 The SGLN consists of the following information elements:

- 481 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the  
 482 same as the Company Prefix digits within a GS1 GLN decimal code.
- 483 • The *Location Reference*, assigned uniquely by the managing entity to an aggregate or  
 484 specific physical location.
- 485 • The *GLN Extension*, assigned by the managing entity to an individual unique location.  
 486 ➤ The use of the GLN Extension is intended for internal purposes. For communication  
 487 between trading partners a GLN will be used. The rules defining the use of the  
 488 SGLN are described in Section 3.7.

489



490

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

492 The SGLN is not explicitly defined in the GS1 General Specifications. However, it may be  
 493 considered equivalent to a GS1-128 bar code that contains both a GLN (Application  
 494 Identifier 414) and an Extension Component (Application Identifier 254). Extension  
 495 Components in AI 254 consist of one to twenty characters, where each character can be a  
 496 digit, uppercase or lowercase letter, or one of a number of allowed punctuation characters.  
 497 The complete set of characters allowed is illustrated in Appendix F. The complete AI 254  
 498 syntax is supported by the pure identity URI syntax specified in Section 4.3.1.

499 When representing Extension Components in 96-bit tags, however, only a subset of the  
 500 Extension Component allowed in the GS1 General Specifications for Application Identifier  
 501 254 is permitted. Specifically, the permitted Extension Component are those consisting of  
 502 one or more digits characters, with no leading zeros, and whose value when considered as an  
 503 integer fits within the range restrictions of the 96-bit tag encodings.

504 While these limitations exist for 96-bit tag encodings, other tag encodings allow a wider  
 505 range of Extension Component. Therefore, application authors and database designers  
 506 should take the GS1 specifications for Application Identifier 254 into account in order to  
 507 accommodate the full range of allowed extension components.

508 For the requirement of using a longer Extension Component, or alphabet and other non  
 509 numeric codings allowed in Application Identifier 254, this specification includes a 195-bit  
 510 tag encoding for SGLN.

511 *Explanation (non-normative): The restrictions are necessary for 96-bit tags in order for the*  
 512 *Extension Component to fit within the small number of bits available in commonly available*  
 513 *96-bit tags. The Extension Component range is restricted to numeric values and an*  
 514 *alphanumeric Extension Component is disallowed. Leading zeros are forbidden so that the*  
 515 *Extension Component can be considered as a decimal integer when encoding the integer*  
 516 *value in binary. By considering it to be a decimal integer, "00034", "034", or "34" (for*  
 517 *example) can't be distinguished as different integer values. In order to insure that every*  
 518 *encoded value can be decoded uniquely, Extension Components can't have leading zeros.*  
 519 *Then, when the bits 0000000000000000000010010 occurs on the tag, the Extension*  
 520 *Component as "34" (not "034" or "00034") is decoded.*

521 .

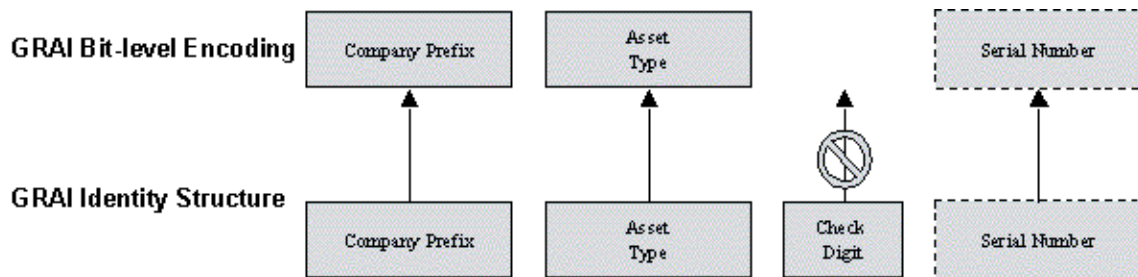
522 **2.1.2.4 Global Returnable Asset Identifier (GRAI)**

523 The Global Returnable Asset Identifier is (GRAI) is defined by the GS1 General  
524 Specifications. Unlike the GTIN, the GRAI is already intended for assignment to individual  
525 objects and therefore does not require any additional fields to serve as an EPC pure identity.

526

527 The GRAI consists of the following information elements:

- 528 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the  
529 same as the Company Prefix digits within a GS1 GRAI decimal code.
- 530 • The *Asset Type*, assigned by the managing entity to a particular class of asset.
- 531 • The *Serial Number*, assigned by the managing entity to an individual object. The GRAI-  
532 96 representation is only capable of representing a subset of Serial Numbers allowed in  
533 the GS1 General Specifications. Specifically, only those Serial Numbers consisting of  
534 one or more digits, with no leading zeros, are permitted [see Appendix E for details].  
535 For the requirement of using longer serial number, or alphabet and other non numeric  
536 codings allowed in Application Identifier 8003, this version of specification includes  
537 longer bit encoding format GRAI-170.



538

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

540 **2.1.2.5 Global Individual Asset Identifier (GIAI)**

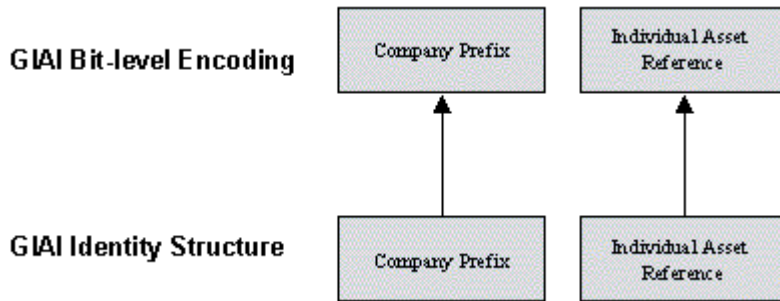
541 The Global Individual Asset Identifier (GIAI) is defined by the GS1 General Specifications.  
542 Unlike the GTIN, the GIAI is already intended for assignment to individual objects and  
543 therefore does not require any additional fields to serve as an EPC pure identity.

544

545 The GIAI consists of the following information elements:

- 546 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the  
547 same as the Company Prefix digits within a GS1 GIAI decimal code.
- 548 • The *Individual Asset Reference*, assigned uniquely by the managing entity to a specific  
549 asset. The GIAI-96 representation is only capable of representing a subset of Individual  
550 Asset References allowed in the GS1 General Specifications. Specifically, only those  
551 Individual Asset References consisting of one or more digits, with no leading zeros, are  
552 permitted.  
553 For the requirement of using longer serial number, or alphabet and other non numeric

554 codings allowed in Application Identifier 8004, this version of specification includes  
 555 the longer bit encoding format GIAI-202.



556  
 557 **Figure G.** How the parts of the decimal GIAI are extracted and rearranged for encoding.

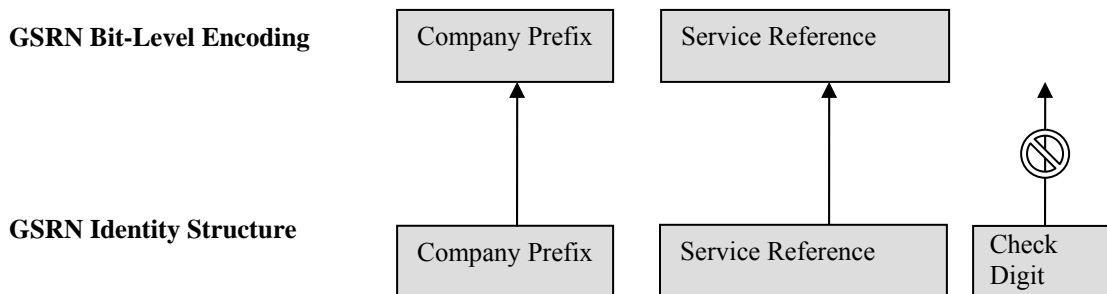
558 **2.1.2.6 Global Service Relation Number (GSRN)**

559 The Global Service Relation Number (GSRN) is defined by the GS1 General Specifications.  
 560 Unlike the GTIN, the GSRN is already intended for assignment to individual objects and  
 561 therefore does not require any additional fields to serve as an EPC pure identity.

562 *Note (Non-Normative): Many applications of GSRN have historically included the*  
 563 *Application Identifier (8018) in the GSRN identifier field when stored in a database. This is*  
 564 *not a standard requirement, but a widespread practice. The Application Identifier is a sort of*  
 565 *header used in bar code applications, and can be inferred directly from EPC headers*  
 566 *representing GSRN. In other words, a GSRN EPC can be interpreted as needed to include*  
 567 *the (8018) as part of the GSRN identifier or not.*

568 The GSRN consists of the following information elements:

- 569 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the  
 570 same as the Company Prefix digits within a GS1 GSRN decimal code.
- 571 • The *Service Reference*, assigned uniquely by the managing entity to identify a specific  
 572 Service Relation. The Service Reference for the purposes of EPC Tag Encoding is  
 573 derived from the GSRN Serial Reference digits, and treating the result as a single  
 574 integer.



576  
 577  
 578 **Figure H.** How the parts of the decimal GSRN are extracted and rearranged for encoding.  
 579

580 **2.1.2.7 Global Document Type Identifier (GDTI)**

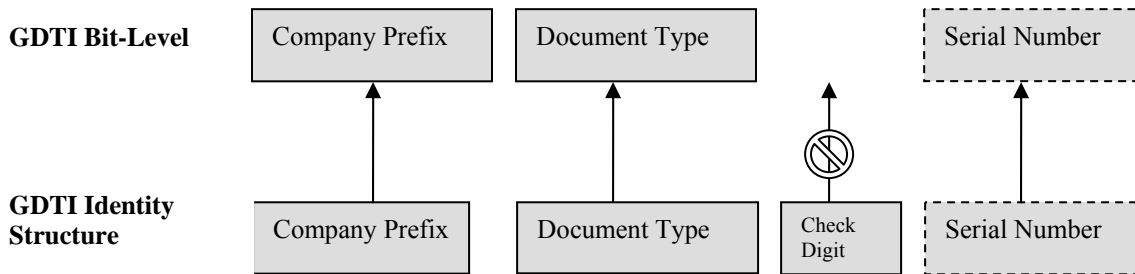
581 The Global Document Type Identifier (GDTI) is defined by the GS1 General Specifications.  
582 Unlike the GTIN, the GDTI is already intended for assignment to individual objects and  
583 therefore does not require any additional fields to serve as an EPC pure identity.

584

585 The GDTI consists of the following information elements:

- 586 • The *Company Prefix*, assigned by GS1 to a managing entity. The Company Prefix is the  
587 same as the Company Prefix digits within a GS1 GRAI decimal code.
- 588 • The *Document Type*, assigned by the managing entity to a particular type of document.
- 589 • The *Serial Number*, optionally assigned by the managing entity to an individual  
590 document. The GDTI-96 representation is only capable of representing a subset of  
591 Serial Numbers allowed in the GS1 General Specifications. Specifically, only those  
592 Serial Numbers consisting of one or more digits, with no leading zeros, are permitted  
593 [see Appendix E for details].  
594 For the requirement of using longer numeric serial numbers, or numeric codings  
595 allowed in Application Identifier 253, this specification includes a 113-bit tag encoding  
596 for GDTI.

597 •



598

599 **Figure I.** How the parts of the decimal GDTI are extracted and rearranged for encoding.

600 **2.1.3 DoD Identity Type**

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

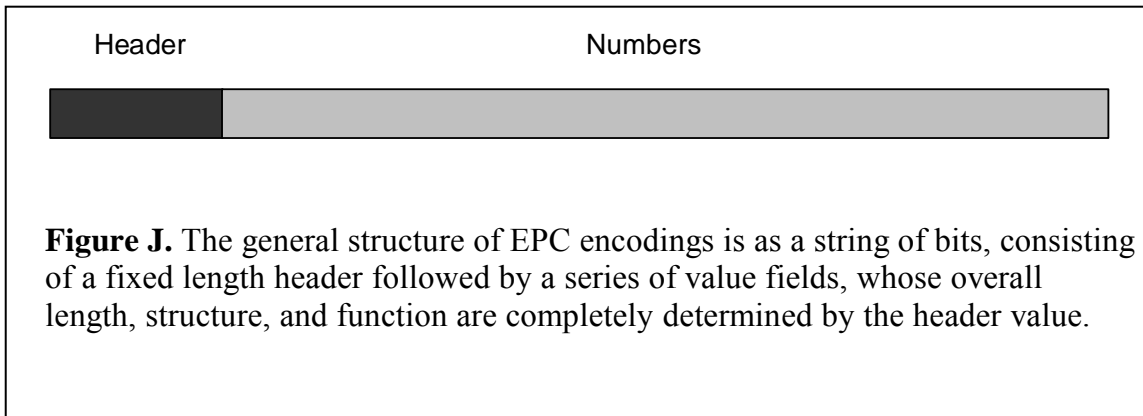
602 This tag data construct may be used to encode 96-bit Class 1 tags for shipping goods to the  
603 United States Department of Defense by a supplier who has already been assigned a CAGE  
604 (Commercial and Government Entity) code.

605 At the time of this writing, the details of what information to encode into these fields is  
606 explained in a document titled "United States Department of Defense Supplier's Passive  
607 RFID Information Guide" that can be obtained at the United States Department of Defense's  
608 website <http://www.dodrfid.org/supplierguide.htm> .

609 **3 EPC Tag Bit-level Encodings**

610 The general structure of EPC Tag Encodings on a tag is as a string of bits (i.e., a binary  
611 representation), consisting of a fixed length (8-bits) header followed by a series of numeric

612 fields (Figure J) whose overall length, structure, and function are completely determined by  
613 the header value. For future expansion purpose, a header value of 11111111 is defined, to  
614 indicate that longer header beyond 8-bits is used.



615

### 616 3.1 Headers

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

623 *Back-compatibility note (non-normative) In a prior version of the Tag Data Standard, the*  
624 *header was of variable length, using a tiered approach in which a zero value in each tier*  
625 *indicated that the header was drawn from the next longer tier. For the encodings defined in*  
626 *the earlier specification, headers were either 2 bits or 8 bits. Given that a zero value is*  
627 *reserved to indicate a header in the next longer tier, the 2-bit header had 3 possible values*  
628 *(01, 10, and 11, not 00), and the 8-bit header had 63 possible values (recognizing that the*  
629 *first 2 bits must be 00 and 00000000 is reserved to allow headers that are longer than 8 bits).*  
630 *The 2-bit headers were only used in conjunction with certain 64-bit EPC Tag Encodings.*

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

638 Fourteen encoding schemes have been defined in this version of the EPC Tag Data Standard,  
639 as shown in Table 1 below. The table also indicates header values that are currently  
640 unassigned, as well as header values that have been reserved to allow for an orderly “sunset”  
641 of 64-bit encodings defined in prior versions of the EPC Tag Data Standard. These will not

642 be available for assignment until after the “sunset date” has passed. The “sunset date” as  
 643 published by EPCglobal July 1, 2006 is July 1, 2009.

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>
<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 1011</u>	<u>10</u> to <u>2B</u>	NA  NA	<u>Reserved for Future Use</u>
0010 1100	2C	96	GDTI-96
0010 1101	2D	96	GSRN-96
0010 1110	2E	96	Reserved for Future Use
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>

Header Value (binary)	Header Value (hex)	Encoding Length (bits)	Encoding Scheme
0011 1000	<u>38</u>	<u>202</u>	<u>GIAI-202</u>
0011 1001	<u>39</u>	<u>195</u>	<u>SGLN-195</u>
0011 1010	<u>3A</u>	<u>113</u>	<u>GDTI-113</u>
<u>0011 1011</u> to <u>0011 1111</u>	<u>3B</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)
<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

644  
645  
646

### 3.2 Use of EPCs on UHF Class 1 Generation 2 Tags

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

650 In the Gen 2 Specification, the tag memory is separated into four distinct banks, each of  
651 which may comprise one or more memory words, where each word is 16 bits long. These  
652 memory banks are described as “Reserved”, “EPC”, “TID” and “User”. The “Reserved”

653 memory bank contains kill and access passwords, the “EPC” memory bank contains data  
654 used for identifying the object to which the tag is or will be attached, the “TID” memory  
655 bank contains data that can be used by the reader to identify the tag’s capability, and “User”  
656 memory bank is intended to contain user-specific data.

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

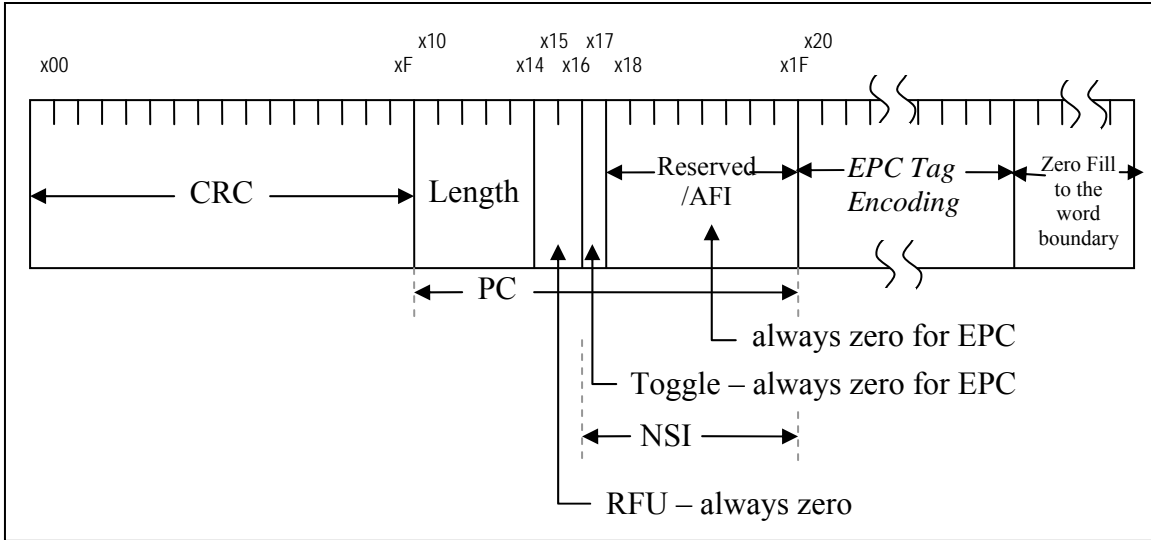
### 663 **3.2.1 EPC Memory Contents**

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

- 666 • *CRC-16 (16 bits)* Bits that represent the error check code and are auto-calculated by the  
667 Tag. (For further details of the CRC, refer to UHF Class 1 Generation 2 Tag Protocol  
668 specification Section 6.3.2.1.3)
- 669 • *Protocol-Control (PC) (16 bits total)* which is subdivided into:
  - 670 • *Length (5 bits)* Represents the number of 16-bit words comprising the PC field and  
671 the EPC field (below). See discussion below for the encoding of this field.
  - 672 • *Reserved for Future Use (RFU) (2 bits)* Always zero in the current version of the  
673 UHF Class 1 Generation 2 Tag Protocol Specification.
  - 674 • *Numbering System Identifier (NSI) (9 bits total)* which is further subdivided into:
    - 675 • *Toggle bit (1 bit)* Boolean flag indicating whether the next 8 bits of the NSI  
676 represents reserved memory or an ISO 15961 Application Family Identifier (AFI).  
677 If set to “zero” indicates that the NSI contains reserved memory, if set to “one”  
678 indicates that the NSI contains an ISO AFI.
    - 679 • *Reserved/AFI (8 bits)* Based on the value of the Toggle Bit above, these 8 bits  
680 are either Reserved and must all be set to zero, or contain an AFI whose value is  
681 defined under the authority of ISO.
- 682 • *EPC (variable length)* When the Toggle Bit is set to zero, an EPC Tag Encoding as  
683 defined in the remaining sections of this chapter is contained here. When the Toggle  
684 Bit is set to “one”, these bits are part of a non-EPC coding scheme identified by the  
685 AFI field (see above) whose interpretation is outside the scope of this specification.
- 686 • *Zero fill (variable length)* If there is any additional memory beyond EPC Tag Encoding  
687 required to meet the 16 bit word boundaries specified in Gen 2 Specification, it is filled  
688 with zeros. An implementation shall not put any data into EPC memory following the  
689 EPC Tag Encoding and any required zero fill (15 bits or less); if it does, it is not in  
690 compliance with the specification and risks the possibility of incompatibility with a  
691 future version of the spec.

692

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



696

697 **Figure K.** Complete contents of EPC memory bank of a Gen 2 Tag.

698

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

701 The complete contents of the EPC are defined by the remaining subsections within this  
702 chapter.

### 703 3.2.2 The Length Bits

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

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

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

719 Moreover, there are applications in which the length field may be set to a different value than  
 720 the one determined by the formula above. For example, there may be applications in which  
 721 the EPC is not written to the EPC bank in one operation, but where a prefix of the EPC is  
 722 written in one operation (perhaps excluding the serial number) and subsequently the  
 723 remainder of the EPC is written. In such an application, a length field smaller than the  
 724 normal value might be used to indicate that the EPC is incompletely written.  
 725

### 726 3.3 Notational Conventions

727 In the remainder of this section, EPC Tag Encoding schemes are depicted using the  
 728 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)

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

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

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

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

- 741 • The value of a binary number indicated by (*Binary value*), as is the case for the  
 742 Header field in the example table above
- 743 • The maximum decimal value indicated by (*Max. decimal value*) of a fixed length  
 744 field. This is calculated as  $2^n - 1$ , where  $n$  = the fixed number of bits in the field.
- 745 • A range of maximum decimal values indicated by (*Max. decimal range*). This range  
 746 is calculated using the normative rules expressed in the related encoding procedure  
 747 section

748 • A reference to a table that provides the valid values defined for the field.

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

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

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

760 **3.4 General Identifier (GID-96)**

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

765

	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)

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

768

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

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

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

778 Number domain. Examples of Object Classes could include case Stock Keeping Units of  
779 consumer-packaged goods and component parts in an assembly.

- 780 • The *Serial Number* code, or serial number, is unique within each object class. In other  
781 words, the managing entity is responsible for assigning unique – non-repeating serial  
782 numbers for every instance within each object class code.

#### 783 **3.4.1.1 GID-96 Encoding Procedure**

784 The following procedure creates a GID-96 encoding.

785 Given:

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

789 Procedure:

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

#### 798 **3.4.1.2 GID-96 Decoding Procedure**

799 Given:

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

802 Yields:

- 803 • A General Manager Number
- 804 • An Object Class
- 805 • A Serial Number

806 Procedure:

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

810 **3.5 Serialized Global Trade Item Number (SGTIN)**

811 The EPC Tag Encoding scheme for SGTIN permits the direct embedding of GS1 System  
 812 standard GTIN and Serial Number codes on EPC tags. In all cases, the check digit is not  
 813 encoded.

814

815 **3.5.1 SGTIN-96**

816 In addition to a Header, the SGTIN-96 is composed of five fields: the *Filter Value*, *Partition*,  
 817 *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)

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

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

- 821 • *Header* is 8-bits, with a binary value of 0011 0000.
- 822 • *Filter Value* is not part of the SGTIN pure identity, but is additional data that is used for  
 823 fast filtering and pre-selection of basic logistics types. The normative specifications  
 824 for Filter Values are specified in Table 5. Values marked as “reserved” are reserved  
 825 for assignment by EPCglobal in future versions of this specification. Implementations  
 826 of the encoding and decoding rules specified below SHALL accept any value of the  
 827 filter bits, whether reserved or not. Applications, however, SHOULD NOT direct an  
 828 encoder to write a reserved value to a tag, nor rely upon a reserved value decoded from  
 829 a tag, as doing so may cause interoperability problems if a reserved value is assigned in  
 830 a future revision to this specification..  
 831 The value of 000 means “All Others”. That is, a filter value of 000 means that the  
 832 object to which the tag is affixed does not match any of the logistic types defined as  
 833 other filter values in this specification. It should be noted that tags conforming to  
 834 earlier versions of this specification, in which 000 was the only value approved for use,  
 835 will have filter value equal to 000, but following the ratification of this standard, the  
 836 filter value should be set to match the object to which the tag is affixed, and use 000  
 837 only if the filter value for such object does not exist in the specification.  
 838 A Standard Trade Item grouping represents all levels of packaging for logistical units.

839  
840  
841

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
Inner Trade Item Grouping not to be sold at Point of Sale	100
Reserved	101
Reserved	110
Reserved	111

842

**Table 5.** SGTIN Filter Values .

843  
844  
845  
846  
847  
848  
849

- *Partition* is an indication of where the subsequent Company Prefix and Item Reference numbers are divided. This organization matches the structure in the GS1 GTIN in which the Company Prefix added to the Item Reference number (prefixed by the single Indicator Digit) totals 13 digits, yet the Company Prefix may vary from 6 to 12 digits and the concatenation of single Indicator Digit and Item Reference from 7 to 1 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Item Reference* fields are defined in Table 6.

850

- *Company Prefix* contains a literal embedding of the GS1 Company Prefix.

851  
852  
853  
854  
855  
856  
857

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

858  
859  
860  
861  
862  
863  
864  
865  
866

- *Serial Number* contains a serial number. The SGTIN-96 encoding is only capable of representing integer-valued serial numbers with limited range. The GS1 specifications permit a broader range of serial numbers. The GS1-128 barcode symbology provides for a 20-character alphanumeric serial number to be associated with a GTIN using Application Identifier (AI) 21 [GS1GS]. It is possible to convert between the serial numbers in the SGTIN-96 tag encoding and the serial numbers in AI 21 barcodes under certain conditions. Specifically, such interconversion is possible when the alphanumeric serial number in AI 21 happens to consist only of digits with no leading zeros, and whose value when interpreted as an integer falls within the range limitations

867 of the SGTIN-96 tag encoding. These considerations are reflected in the encoding and  
 868 decoding procedures below.

869

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
0	40	12	4	1
1	37	11	7	2
2	34	10	10	3
3	30	9	14	4
4	27	8	17	5
5	24	7	20	6
6	20	6	24	7

870 **Table 6.** SGTIN Partitions.

871 **3.5.1.1 SGTIN-96 Encoding Procedure**

872 The following procedure creates an SGTIN-96 encoding.

873 Given:

- 874 • A GS1 GTIN-14 consisting of digits  $d_1d_2\dots d_{14}$
- 875 • The length  $L$  of the Company Prefix portion of the GTIN
- 876 • A Serial Number  $S$  where  $0 \leq S < 2^{38}$ , or a GS1-128 Application Identifier 21 consisting  
 877 of characters  $s_1s_2\dots s_K$ .
- 878 • A Filter Value  $F$  where  $0 \leq F < 8$

879 Procedure:

- 880 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of  
 881 the Partition Table (Table 6) to determine the Partition Value,  $P$ , the number of bits  $M$  in the  
 882 Company Prefix field, and the number of bits  $N$  in the Item Reference and Indicator Digit  
 883 field. If  $L$  is not found in any row of Table 6, stop: this GTIN cannot be encoded in an  
 884 SGTIN-96.
- 885 2. Construct the Company Prefix by concatenating digits  $d_2d_3\dots d_{(L+1)}$  and considering the  
 886 result to be a decimal integer,  $C$ .
- 887 3. Construct the Indicator Digit and Item Reference by concatenating digits  
 888  $d_1d_{(L+2)}d_{(L+3)}\dots d_{13}$  and considering the result to be a decimal integer,  $I$ .

889 4. When the Serial Number is provided directly as an integer  $S$  where  $0 \leq S < 2^{38}$ , proceed to  
 890 Step 5. Otherwise, when the Serial Number is provided as a GS1-128 Application Identifier  
 891 21 consisting of characters  $s_1s_2\dots s_K$ , construct the Serial Number by concatenating digits  
 892  $s_1s_2\dots s_K$ . If any of these characters is not a digit, stop: this Serial Number cannot be  
 893 encoded in the SGTIN-96 encoding. Also, if  $K > 1$  and  $s_1 = 0$ , stop: this Serial Number  
 894 cannot be encoded in the SGTIN-96 encoding (because leading zeros are not permitted  
 895 except in the case where the Serial Number consists of a single zero digit). Otherwise,  
 896 consider the result to be a decimal integer,  $S$ . If  $S \geq 2^{38}$ , stop: this Serial Number cannot be  
 897 encoded in the SGTIN-96 encoding.

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

### 902 3.5.1.2 SGTIN-96 Decoding Procedure

903 Given:

- 904 • An SGTIN-96 as a 96-bit bit string  $00110000b_{87}b_{86}\dots b_0$  (where the first eight bits  
 905  $00110000$  are the header)

906 Yields:

- 907 • A GS1 GTIN-14
- 908 • A Serial Number
- 909 • A Filter Value

910 Procedure:

- 911 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 912 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
 913  $P = 7$ , stop: this bit string cannot be decoded as an SGTIN-96.
- 914 3. Look up the Partition Value  $P$  in Table 6 to obtain the number of bits  $M$  in the Company  
 915 Prefix and the number of digits  $L$  in the Company Prefix.
- 916 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned integer.  
 917 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal SGTIN-  
 918 96 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ , adding  
 919 leading zeros as necessary to make up  $L$  digits in total.
- 920 5. Extract the Item Reference and Indicator by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{38}$  as an  
 921 unsigned integer. If this integer is greater than or equal to  $10^{(13-L)}$ , stop: the input bit string  
 922 is not a legal SGTIN-96 encoding. Otherwise, convert this integer to a (13-L)-digit decimal  
 923 number  $i_1i_2\dots i_{(13-L)}$ , adding leading zeros as necessary to make (13-L) digits.
- 924 6. Construct a 13-digit number  $d_1d_2\dots d_{13}$  where  $d_1 = i_1$  from Step 5,  $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$   
 925 from Step 4, and  $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2i_3\dots i_{(13-L)}$  from Step 5.

- 926 7. Calculate the check digit  $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8 +$   
 927  $d_{10} + d_{12})) \bmod 10$ .
- 928 8. The GS1 GTIN-14 is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{14}$ .
- 929 9. Bits  $b_{37}b_{36}\dots b_0$ , considered as an unsigned integer, are the Serial Number.
- 930 10. (Optional) If it is desired to represent the serial number as a GS1-128 Application  
 931 Identifier 21, convert the integer from Step 9 to a decimal string with no leading zeros. If the  
 932 integer in Step 9 is zero, convert it to a string consisting of the single character “0”.

933 **3.5.2 SGTIN-198**

934 In addition to a Header, the SGTIN-198 is composed of five fields: the *Filter Value*,  
 935 *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

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

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

- 939 • *Header* is 8-bits, with a binary value of 0011 0110.
- 940 • *Filter Value* is not part of the GTIN or EPC identifier, but is used for fast filtering and  
 941 pre-selection of basic logistics types. The normative Filter Values for 96-bit and 198-  
 942 bit GTIN are specified in Table 5. Values marked as “reserved” are reserved for  
 943 assignment by EPCglobal in future versions of this specification. Implementations of  
 944 the encoding and decoding rules specified below SHALL accept any value of the filter  
 945 bits, whether reserved or not. Applications, however, SHOULD NOT direct an  
 946 encoder to write a reserved value to a tag, nor rely upon a reserved value decoded from  
 947 a tag, as doing so may cause interoperability problems if a reserved value is assigned in  
 948 a future revision to this specification.
- 949 • *Partition* is an indication of where the subsequent Company Prefix and Item Reference  
 950 numbers are divided. This organization matches the structure in the GS1 GTIN in  
 951 which the Company Prefix added to the Item Reference number (prefixed by the single  
 952 Indicator Digit) totals 13 digits, yet the Company Prefix may vary from 6 to 12 digits  
 953 and the Item Reference (including the single Indicator Digit) from 7 to 1 digit(s). The

954 available values of *Partition* and the corresponding sizes of the *Company Prefix* and  
955 *Item Reference* fields are defined in Table 6.

- 956 • *Company Prefix* contains a literal embedding of the GS1 Company Prefix.
- 957 • *Item Reference* contains a literal embedding of the GTIN Item Reference number. The  
958 Indicator Digit is combined with the Item Reference field in the following manner:  
959 Leading zeros on the item reference are significant. Put the Indicator Digit in the  
960 leftmost position available within the field. *For instance, 00235 is different than 235.*  
961 *With the indicator digit of 1, the combination with 00235 is 100235.* The resulting  
962 combination is treated as a single integer, and encoded into binary to form the *Item*  
963 *Reference* field.
- 964 • *Serial Number* contains a serial number. The SGTIN-198 encoding is capable of  
965 representing alphanumeric serial numbers of up to 20 characters, permitting the full  
966 range of serial numbers available in the GS1-128 barcode symbology using  
967 Application Identifier (AI) 21 [GS1GS]. See Appendix F for permitted values.

968

### 969 3.5.2.1 SGTIN-198 Encoding Procedure

970 The following procedure creates an SGTIN-198 encoding.

971 Given:

- 972 • A GS1 GTIN-14 consisting of digits  $d_1d_2\dots d_{14}$
- 973 • The length  $L$  of the Company Prefix portion of the GTIN
- 974 • A GS1-128 Application Identifier 21 consisting of characters  $s_1s_2\dots s_K$ , where  $K \leq 20$ .
- 975 • A Filter Value  $F$  where  $0 \leq F < 8$

976 Procedure:

- 977 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of  
978 the Partition Table (Table 6) to determine the Partition Value,  $P$ , the number of bits  $M$  in the  
979 Company Prefix field, and the number of bits  $N$  in the Item Reference and Indicator Digit  
980 field. If  $L$  is not found in any row of Table 6, stop: this GTIN cannot be encoded in an  
981 SGTIN-198.
- 982 2. Construct the Company Prefix by concatenating digits  $d_2d_3\dots d_{(L+1)}$  and considering the  
983 result to be a decimal integer,  $C$ .
- 984 3. Construct the Indicator Digit and Item Reference by concatenating digits  
985  $d_1d_{(L+2)}d_{(L+3)}\dots d_{13}$  and considering the result to be a decimal integer,  $I$ .
- 986 4. Check that each of the characters  $s_1s_2\dots s_K$  is one of the 82 characters listed in the table  
987 in Appendix F. If this is not the case, stop: this character string cannot be encoded as an  
988 SGTIN-198. Otherwise construct the Serial Number by concatenating the 7-bit code, as  
989 given in Appendix F, for each of the characters  $s_1s_2\dots s_K$ , yielding  $7K$  bits total. If  $K < 20$ ,  
990 concatenate additional zero bits to the right to make a total of 140 bits.

991 5. Construct the final encoding by concatenating the following bit fields, from most  
 992 significant to least significant: Header 00110110 (8 bits), Filter Value  $F$  (3 bits), Partition  
 993 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Item Reference from  
 994 Step 3 ( $N$  bits) and Serial Number from Step 4 (140 bits). Note that  $M+N = 44$  bits for all  $P$ .

### 995 3.5.2.2 SGTIN-198 Decoding Procedure

996 Given:

997 • An SGTIN-198 as a 198-bit bit string  $00110110b_{189}b_{188}\dots b_0$  (where the first eight bits  
 998  $00110110$  are the header)

999 Yields:

1000 • A GS1 GTIN-14

1001 • A Serial Number

1002 • A Filter Value

1003 Procedure:

1004 1. Bits  $b_{189}b_{188}b_{187}$ , considered as an unsigned integer, are the Filter Value.

1005 2. Extract the Partition Value  $P$  by considering bits  $b_{186}b_{185}b_{184}$  as an unsigned integer. If  
 1006  $P = 7$ , stop: this bit string cannot be decoded as an SGTIN-198.

1007 3. Look up the Partition Value  $P$  in Table 6 to obtain the number of bits  $M$  in the Company  
 1008 Prefix and the number of digits  $L$  in the Company Prefix.

1009 4. Extract the Company Prefix  $C$  by considering bits  $b_{183}b_{182}\dots b_{(184-M)}$  as an unsigned  
 1010 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal  
 1011 SGTIN-198 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ ,  
 1012 adding leading zeros as necessary to make up  $L$  digits in total.

1013 5. Extract the Item Reference and Indicator by considering bits  $b_{(183-M)}b_{(182-M)}\dots b_{140}$  as an  
 1014 unsigned integer. If this integer is greater than or equal to  $10^{(13-L)}$ , stop: the input bit string  
 1015 is not a legal SGTIN-198 encoding. Otherwise, convert this integer to a  $(13-L)$ -digit decimal  
 1016 number  $i_1i_2\dots i_{(13-L)}$ , adding leading zeros as necessary to make  $(13-L)$  digits.

1017 6. Construct a 13-digit number  $d_1d_2\dots d_{13}$  where  $d_1 = i_1$  from Step 5,  $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$   
 1018 from Step 4, and  $d_{(L+2)}d_{(L+3)}\dots d_{13} = i_2i_3\dots i_{(13-L)}$  from Step 5.

1019 7. Calculate the check digit  $d_{14} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13}) - (d_2 + d_4 + d_6 + d_8 +$   
 1020  $d_{10} + d_{12})) \bmod 10$ .

1021 8. The GS1 GTIN-14 is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{14}$ .

1022 9. Divide the remaining bits  $b_{139}b_{138}\dots b_0$  into 7-bit segments. The result should consist of  $K$   
 1023 non-zero segments followed by  $20-K$  zero segments. If this is not the case, stop: this bit  
 1024 string cannot be decoded as an SGTIN-198. Otherwise, look up each of the non-zero 7-bit  
 1025 segments in Appendix F to obtain a corresponding character. If any of the non-zero 7-bit  
 1026 segments has a value that is not in Appendix F, stop: this bit string cannot be decoded as an  
 1027 SGTIN-198. Otherwise, the  $K$  characters so obtained, considered as a character string, is the  
 1028 value of the GS1 AI 21.

1029 10. The GS1 SGTIN-198 is the concatenation of the digits from Steps 6 and 7 and the  
 1030 characters from Step 9. :  $d_1d_2\dots d_{14} s_1s_2\dots s_K$

1031

1032

### 1033 3.6 Serial Shipping Container Code (SSCC)

1034 The EPC Tag Encoding scheme for SSCC permits the direct embedding of GS1 System  
 1035 standard SSCC codes on EPC tags. In all cases, the check digit is not encoded.

#### 1036 3.6.1 SSCC-96

1037 In addition to a Header, the EPC SSCC-96 is composed of four fields: the *Filter Value*,  
 1038 *Partition*, *Company Prefix*, and *Serial Reference*, as shown in Table 8.

1039

	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 – 99,999 (Max. decimal range*)	[Not Used]

1040 \*Max. decimal value range of Company Prefix and Serial Reference fields vary according to the contents of the  
 1041 Partition field.

1042 **Table 8.** The EPC 96-bit SSCC bit allocation, header, and maximum decimal values.

1043 • *Header* is 8-bits, with a binary value of 0011 0001.

1044 • *Filter Value* is not part of the SSCC or EPC identifier, but is used for fast filtering and  
 1045 pre-selection of basic logistics types. The normative specifications for Filter Values  
 1046 are specified in Table 9. Values marked as “reserved” are reserved for assignment by  
 1047 EPCglobal in future versions of this specification. Implementations of the encoding  
 1048 and decoding rules specified below SHALL accept any value of the filter bits, whether  
 1049 reserved or not. Applications, however, SHOULD NOT direct an encoder to write a  
 1050 reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so  
 1051 may cause interoperability problems if a reserved value is assigned in a future revision  
 1052 to this specification.

1053 The value of 000 means “All Others”. That is, a filter value of 000 means that the  
 1054 object to which the tag is affixed does not match any of the logistic types defined as  
 1055 other filter values in the specification. It should be noted that tags conforming to earlier

1056  
1057  
1058  
1059

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

1060

**Table 9.** SSCC Filter Values

1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068

- The *Partition* is an indication of where the subsequent Company Prefix and Serial Reference numbers are divided. This organization matches the structure in the GS1 SSCC-18 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 (P)	Company Prefix		Extension Digit and Serial Reference	
	Bits (M)	Digits (L)	Bits (N)	Digits
0	40	12	18	5
1	37	11	21	6
2	34	10	24	7
3	30	9	28	8
4	27	8	31	9
5	24	7	34	10
6	20	6	38	11

1069

**Table 10.** SSCC-96 Partitions.

1070

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

1071

- *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 GS1 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.

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

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

1083

### **3.6.1.1 SSCC-96 Encoding Procedure**

1084

The following procedure creates an SSCC-96 encoding.

1085

Given:

1086

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

1087

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

1088

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

1089

Procedure:

1090

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

1091

1092

1093

1094

1095

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

1096

1097

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

1098

1099

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

1100

1101

1102

1103

### **3.6.1.2 SSCC-96 Decoding Procedure**

1104

Given:

1105 • An SSCC-96 as a 96-bit bit string  $00110001b_{87}b_{86}\dots b_0$  (where the first eight bits  
1106  $00110001$  are the header)

1107 Yields:

1108 • An SSCC-18

1109 • A Filter Value

1110 Procedure:

1111 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.

1112 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
1113  $P = 7$ , stop: this bit string cannot be decoded as an SSCC-96.

1114 3. Look up the Partition Value  $P$  in Table 10 to obtain the number of bits  $M$  in the Company  
1115 Prefix and the number of digits  $L$  in the Company Prefix.

1116 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned integer.  
1117 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal SSCC-96  
1118 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ , adding leading  
1119 zeros as necessary to make up  $L$  digits in total.

1120 5. Extract the Serial Reference by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{24}$  as an unsigned integer.  
1121 If this integer is greater than or equal to  $10^{(17-L)}$ , stop: the input bit string is not a legal  
1122 SSCC-96 encoding. Otherwise, convert this integer to a  $(17-L)$ -digit decimal number  
1123  $i_1i_2\dots i_{(17-L)}$ , adding leading zeros as necessary to make  $(17-L)$  digits.

1124 6. Construct a 17-digit number  $d_1d_2\dots d_{17}$  where  $d_1 = s_1$  from Step 5,  $d_2d_3\dots d_{(L+1)} = p_1p_2\dots p_L$   
1125 from Step 4, and  $d_{(L+2)}d_{(L+3)}\dots d_{17} = i_2i_3\dots i_{(17-L)}$  from Step 5.

1126 7. Calculate the check digit  $d_{18} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13} + d_{15} + d_{17}) - (d_2 + d_4$   
1127  $+ d_6 + d_8 + d_{10} + d_{12} + d_{14} + d_{16})) \bmod 10$ .

1128 8. The SSCC-18 is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{18}$ .

### 1129 **3.7 Serialized Global Location Number (SGLN)**

1130 The EPC Tag Encoding scheme for GLN permits the direct embedding of the GS1 System  
1131 standard GLN on EPC tags. GS1 has defined the GLN as AI (414) and has defined a GLN  
1132 Extension Component as AI (254). The AI (254) uses the Set of Characters defined in  
1133 Appendix F.

1134 The use of the GLN Extension Component is intended for internal company purposes. For  
1135 communication between trading partners a GLN will be used. Trading partners can only use  
1136 the GLN Extension through mutual agreement but would have to establish an “out of band”  
1137 exchange of master data describing the extensions. If the GLN only encoding is used, then  
1138 the *Extension Component* shall be set to a fixed value of binary “0” for SGLN-96 and to  
1139 binary 0110000 followed by 133 binary “0” bits for SGLN-195 encoding as described in the  
1140 following SGLN procedures. In all cases the check digit is not encoded.

1141 **3.7.1 SGLN-96**

1142 In addition to a Header, the SGLN-96 is composed of five fields: the *Filter Value*, *Partition*,  
 1143 *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.

1144 \*Max. decimal value range of Company Prefix and Location Reference fields vary according to contents of the  
 1145 Partition field.

1146 **Table 11.** The EPC SGLN-96 bit allocation, header, and maximum decimal values.

- 1147 • *Header* is 8-bits, with a binary value of 0011 0010.
- 1148 • *Filter Value* is not part of the GLN or EPC identifier, but is used for fast filtering and  
 1149 pre-selection of basic location types. The Filter Values for an SGLN-96 is shown in  
 1150 Table 12 below. Values marked as “reserved” are reserved for assignment by  
 1151 EPCglobal in future versions of this specification. Implementations of the encoding  
 1152 and decoding rules specified below SHALL accept any value of the filter bits, whether  
 1153 reserved or not. Applications, however, SHOULD NOT direct an encoder to write a  
 1154 reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so  
 1155 may cause interoperability problems if a reserved value is assigned in a future revision  
 1156 to this specification.

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

Type	Binary Value
Reserved	101
Reserved	110
Reserved	111

**Table 12.** SGLN Filter Values.

1157

1158

1159

- *Partition* is an indication of where the subsequent Company Prefix and Location Reference numbers are divided. This organization matches the structure in the GS1 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.

1166

- *Company Prefix* contains a literal embedding of the GS1 Company Prefix.

1167

- *Location Reference*, if present, encodes the GLN Location Reference number.

1168

- *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 GS1 specifications permit a broader range of Extension Components. The GS1-128 barcode symbology provides for a 20-character alphanumeric Extension Component to be associated with a GLN using Application Identifier (AI) 254 [GS1GS]. 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 SGLN-96 tag encoding. These considerations are reflected in the encoding and decoding procedures below.

1181

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.

1182

1183 **3.7.1.1 SGLN-96 Encoding Procedure**

1184 The following procedure creates an SGLN-96 encoding.

1185 Given:

- 1186 • A GS1 GLN consisting of digits  $d_1d_2\dots d_{13}$
- 1187 • The length  $L$  of the Company Prefix portion of the GLN
- 1188 • An Extension Component  $S$  where  $0 \leq S < 2^{40}$ , or a GS1-128 Application Identifier 254
- 1189 consisting of characters  $s_1s_2\dots s_K$ , When the Extension Component  $S$  is 0, the Encoding
- 1190 will be considered as a GLN only.
- 1191
- 1192 • A Filter Value  $F$  where  $0 \leq F < 8$

1193 Procedure:

- 1194 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of
- 1195 the Partition Table (Table 13) to determine the Partition Value,  $P$ , the number of bits  $M$  in
- 1196 the Company Prefix field, and the number of bits  $N$  in the Location Reference field. If  $L$  is
- 1197 not found in any row of Table 13, stop: this GLN cannot be encoded in an SGLN-96.
- 1198 2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_L$  and considering the result
- 1199 to be a decimal integer,  $C$ .
- 1200 3. If  $L < 12$  construct the Location Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and
- 1201 considering the result to be a decimal integer,  $I$ . If  $L = 12$  set  $b_{41}$  to 0 since there is no
- 1202 Location Reference digit.
- 1203 4. When the Extension Component is provided directly as an integer  $S$  where  $0 \leq S < 2^{40}$ ,
- 1204 proceed to Step 5. Otherwise, when the Extension Component is provided as a GS1-128
- 1205 Application Identifier 254 consisting of characters  $s_1s_2\dots s_K$ , construct the Extension
- 1206 Component by concatenating characters  $s_1s_2\dots s_K$ . If any of these characters is not a digit,
- 1207 stop: this Extension Component cannot be encoded in the SGLN-96 encoding. Also, if  $K >$
- 1208  $1$  and  $s_1 = 0$ , stop: this Extension Component cannot be encoded in the SGLN-96 encoding
- 1209 (because leading zeros are not permitted except in the case where the Extension Component
- 1210 consists of a single zero digit). Otherwise, consider the result to be a decimal integer,  $S$ . If  $S$
- 1211  $\geq 2^{40}$ , stop: this Extension Component cannot be encoded in the SGLN-96 encoding.
- 1212 5. Construct the final encoding by concatenating the following bit fields, from most
- 1213 significant to least significant: Header 00110010 (8 bits), Filter Value  $F$  (3 bits), Partition
- 1214 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Location Reference  $I$

1215 from Step 3 ( $N$  bits) and Extension Component  $S$  from Step 4 (41 bits). Note that  $M+N =$   
1216 41 bits for all  $P$ .

### 1217 **3.7.1.2 SGLN-96 Decoding Procedure**

1218 Given:

1219 • An SGLN-96 as a 96-bit bit string  $00110010b_{87}b_{86}\dots b_0$  (where the first eight bits  
1220  $00110010$  are the header)

1221 Yields:

- 1222 • A GS1 GLN
- 1223 • An Extension Component
- 1224 • A Filter Value

1225 Procedure:

- 1226 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1227 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
1228  $P = 7$ , stop: this bit string cannot be decoded as an SGLN-96.
- 1229 3. Look up the Partition Value  $P$  in Table 13 to obtain the number of bits  $M$  in the Company  
1230 Prefix and the number of digits  $L$  in the Company Prefix.
- 1231 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned integer.  
1232 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal SGLN-96  
1233 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ , adding leading  
1234 zeros as necessary to make up  $L$  digits in total.
- 1235 5. If  $L < 12$  extract the Location Reference by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{41}$  as an  
1236 unsigned integer. If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit string  
1237 is not a legal SGLN-96 encoding. Otherwise, convert this integer to a  $(12-L)$ -digit decimal  
1238 number  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as necessary to make  $(12-L)$  digits.
- 1239 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  $L <$   
1240  $12$   $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_1 i_2\dots i_{(12-L)}$  from Step 5.
- 1241 7. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$   
1242  $d_{11})) \bmod 10$ .
- 1243 8. The GS1 GLN is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{13}$ .
- 1244 9. Bits  $b_{40}b_{39}\dots b_0$ , considered as an unsigned integer, are the *Extension Component*.
- 1245 10. (Optional) If it is desired to represent the Extension Component as a GS1-128  
1246 Application Identifier 254, convert the integer from Step 9 to a decimal string with no  
1247 leading zeros. If the integer in Step 9 is zero, convert it to a string consisting of the single  
1248 character "0".

1249 **3.7.2 SGLN-195**

1250 In addition to a Header, the SGLN-195 is composed of five fields: the *Filter Value*, *Partition*,  
 1251 *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 – 999,999,999,999 (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.

1252 \*Max. decimal value range of Company Prefix and Location Reference fields vary according to contents of the  
 1253 Partition field.

1254 **Table 14.** The EPC SGLN-195 bit allocation, header, and maximum decimal values.

- 1255 • *Header* is 8-bits, with a binary value of 0011 1001.
- 1256 • *Filter Value* is not part of the GLN or EPC identifier, but is used for fast filtering and  
 1257 pre-selection of basic location types. The Filter Values for an SGLN-195 is shown in  
 1258 Table 12. Values marked as “reserved” are reserved for assignment by EPCglobal in  
 1259 future versions of this specification. Implementations of the encoding and decoding  
 1260 rules specified below SHALL accept any value of the filter bits, whether reserved or  
 1261 not. Applications, however, SHOULD NOT direct an encoder to write a reserved  
 1262 value to a tag, nor rely upon a reserved value decoded from a tag, as doing so may  
 1263 cause interoperability problems if a reserved value is assigned in a future revision to  
 1264 this specification
- 1265 • *Partition* is an indication of where the subsequent Company Prefix and Location  
 1266 Reference numbers are divided. This organization matches the structure in the GS1  
 1267 GLN in which the Company Prefix added to the Location Reference number totals 12  
 1268 digits, yet the Company Prefix may vary from 6 to 12 digits and the Location  
 1269 Reference number from 6 to 0 digit(s). The available values of *Partition* and the  
 1270 corresponding sizes of the *Company Prefix* and *Location Reference* fields are defined  
 1271 in Table 13.
- 1272 • *Company Prefix* contains a literal embedding of the GS1 Company Prefix.
- 1273 • *Location Reference*, if present, encodes the GLN Location Reference number.
- 1274 • *Extension Component* contains a serial number. If an *Extension Component* is not used  
 1275 signifying a GLN only, then this value shall be set to binary 0110000 followed by 133  
 1276 binary “0” bits. SGLN.-195 encoding is capable of representing alphanumeric  
 1277 Extension Component of up to 20 characters, permitting the full range of Extension

1278 Component available in the GS1-128 barcode symbology using Application Identifier  
1279 (AI) 254 [GS1GS]. See Appendix F for permitted values.

### 1280 **3.7.2.1 SGLN-195 Encoding Procedure**

1281 The following procedure creates an SGLN-195 encoding.

1282 Given:

- 1283 • A GS1 GLN consisting of digits  $d_1d_2\dots d_{13}$
- 1284 • The length  $L$  of the Company Prefix portion of the GLN
- 1285 • A GS1-128 Application Identifier 254 consisting of characters  $s_1s_2\dots s_K$ , where  $K \leq 20$ .  
1286 If the Application Identifier 254 consists of a single character 0 where  $K=1$ , this  
1287 Encoding is considered to be a GLN only.
- 1288 • A Filter Value  $F$  where  $0 \leq F < 8$

1289 Procedure:

- 1290 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of  
1291 the Partition Table (Table 13) to determine the Partition Value,  $P$ , the number of bits  $M$  in  
1292 the Company Prefix field, and the number of bits  $N$  in the Location Reference field. If  $L$  is  
1293 not found in any row of Table 13, stop: this GLN cannot be encoded in an SGLN-195.
- 1294 2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_L$  and considering the result  
1295 to be a decimal integer,  $C$ .
- 1296 3. If  $L < 12$  construct the Location Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and  
1297 considering the result to be a decimal integer,  $I$ . If  $L = 12$  set  $b_{140}$  to 0 since there is no  
1298 Location Reference digit.
- 1299 4. Check that each of the characters  $s_1s_2\dots s_K$  is one of the 82 characters listed in the table  
1300 in Appendix F. If this is not the case, stop: this character string cannot be encoded as an  
1301 SGLN-195. Otherwise construct the Extension Component by concatenating the 7-bit code,  
1302 as given in Appendix F, for each of the characters  $s_1s_2\dots s_K$ , yielding  $7K$  bits total. If  $K < 20$ ,  
1303 concatenate additional zero bits to the right to make a total of 140 bits.
- 1304 5. Construct the final encoding by concatenating the following bit fields, from most  
1305 significant to least significant: Header 00111001 (8 bits), Filter Value  $F$  (3 bits), Partition  
1306 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Location Reference  $I$   
1307 from Step 3 ( $N$  bits) and Extension Component  $S$  from Step 4 (140 bits). Note that  $M+N =$   
1308 41 bits for all  $P$ .

### 1309 **3.7.2.2 SGLN-195 Decoding Procedure**

1310 Given:

- 1311 • An SGLN-195 as a 195-bit bit string  $00111001b_{186}b_{185}\dots b_0$  (where the first eight bits  
1312  $00111001$  are the header)

1313 Yields:

- 1314 • A GS1 GLN

- 1315     • An Extension Component
- 1316     • A Filter Value
- 1317 Procedure:
- 1318 1. Bits  $b_{186}b_{185}b_{184}$ , considered as an unsigned integer, are the Filter Value.
- 1319 2. Extract the Partition Value  $P$  by considering bits  $b_{183}b_{182}b_{181}$  as an unsigned integer. If
- 1320  $P = 7$ , stop: this bit string cannot be decoded as an SGLN-195.
- 1321 3. Look up the Partition Value  $P$  in Table 13 to obtain the number of bits  $M$  in the Company
- 1322 Prefix and the number of digits  $L$  in the Company Prefix.
- 1323 4. Extract the Company Prefix  $C$  by considering bits  $b_{180}b_{179}\dots b_{(181-M)}$  as an unsigned
- 1324 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal
- 1325 SGLN-195 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ ,
- 1326 adding leading zeros as necessary to make up  $L$  digits in total.
- 1327 5. When  $L < 12$  extract the Location Reference by considering bits  $b_{(180-M)}b_{(179-M)}\dots b_{140}$  as
- 1328 an unsigned integer. If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit
- 1329 string is not a legal SGLN-195 encoding. Otherwise, convert this integer to a  $(12-L)$ -digit
- 1330 decimal number  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as necessary to make  $(12-L)$  digits.
- 1331 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  $L <$
- 1332  $12$   $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_2i_3\dots i_{(12-L)}$  from Step 5.
- 1333 7. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$
- 1334  $d_{11})) \bmod 10$ .
- 1335 8. The GS1 GLN is the concatenation of digits from Steps 6 and 7:  $d_1d_2\dots d_{13}$ .
- 1336 9. Divide the remaining bits  $b_{139}b_{138}\dots b_0$  into 7-bit segments. The result should consist of  $K$
- 1337 non-zero binary segments followed by  $20-K$  binary zero segments. If this is not the case,
- 1338 stop: this bit string cannot be decoded as an SGLN-195. Otherwise, look up each of the
- 1339 non-zero 7-bit segments in Appendix F to obtain a corresponding character. If any of the
- 1340 non-zero 7-bit segments has a value that is not in Appendix F, stop: this bit string cannot be
- 1341 decoded as an SGLN-195. If  $K=1$  and  $s_1=0$ , then this indicates a GLN only with no
- 1342 *Extension Component*. Otherwise, the  $K$  characters so obtained, considered as a character
- 1343 string  $s_1s_2\dots s_K$ , is the value of the GS1 AI 254.
- 1344 10. The GS1 SGLN-195 is the concatenation of the digits from Steps 6 and 7 and the
- 1345 characters from Step 9. :  $d_1d_2\dots d_{13} s_1s_2\dots s_K$

1346

### 1347 **3.8 Global Returnable Asset Identifier (GRAI)**

1348 The EPC Tag Encoding scheme for GRAI permits the direct embedding of a GS1 System

1349 standard GRAI on EPC tags. In all cases, the check digit is not encoded. Only GRAIs that

1350 include the optional serial number may be represented as EPCs. A GRAI without a serial

1351 number represents an asset class, rather than a specific instance, and therefore may not be

1352 used as an EPC (just as a non-serialized GTIN may not be used as an EPC).

1353 *Explanation (non-normative): In the specification of the encoding and decoding procedures*  
 1354 *below, a GS1 GRAI is shown consisting of a 13-digit code (including check digit) together*  
 1355 *with a variable-length serial number. When a GRAI is encoded into a GS1-128 bar code*  
 1356 *using AI 8002, an extra zero digit is prepended to the GRAI. This leading zero is not shown*  
 1357 *in the encoding and decoding procedures. The digit  $d_1$  in the encoding and decoding*  
 1358 *procedures below corresponds to digit N1 in the GS1 General Specifications. Sections*  
 1359 *2.3.3.1.1 and 3.6.50.*

1360

1361 **3.8.1 GRAI-96**

1362 In addition to a Header, the GRAI-96 is composed of five fields: the *Filter Value*, *Partition*,  
 1363 *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)

1364 \*Max. decimal value range of Company Prefix and Asset Type fields vary according to contents of the Partition  
 1365 field.

1366 **Table 15.** The EPC GRAI-96 bit allocation, header, and maximum decimal values.

- 1367 • *Header* is 8-bits, with a binary value of 0011 0011.
- 1368 • *Filter Value* is not part of the GRAI or EPC identifier, but is used for fast filtering and  
 1369 pre-selection of basic asset types. The Filter Values for 96-bit and 170-bit GRAI are  
 1370 shown in Table 16. Values marked as “reserved” are reserved for assignment by  
 1371 EPCglobal in future versions of this specification. Implementations of the encoding  
 1372 and decoding rules specified below SHALL accept any value of the filter bits, whether  
 1373 reserved or not. Applications, however, SHOULD NOT direct an encoder to write a  
 1374 reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so  
 1375 may cause interoperability problems if a reserved value is assigned in a future revision  
 1376 to this specification.

Type	Binary Value
All Others	000
Reserved	001

Type	Binary Value
Reserved	010
Reserved	011
Reserved	100
Reserved	101
Reserved	110
Reserved	111

**Table 16.** GRAI Filter Values

1377  
1378  
1379  
1380  
1381  
1382  
1383

- *Partition* is an indication of where the subsequent Company Prefix and Asset Type numbers are divided. This organization matches the structure in the GS1 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 (P)	Company Prefix		Asset Type	
	Bits (M)	Digits (L)	Bits (N)	Digits
0	40	12	4	0
1	37	11	7	1
2	34	10	10	2
3	30	9	14	3
4	27	8	17	4
5	24	7	20	5
6	20	6	24	6

**Table 17.** GRAI Partitions.

1384  
1385  
1386  
1387  
1388  
1389  
1390

- *Company Prefix* contains a literal embedding of the GS1 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 GS1 General Specifications. The capacity of this mandatory serial number is less than the maximum GS1 System

1391 specification for serial number, no leading zeros are permitted, and only numbers are  
1392 permitted.

### 1393 **3.8.1.1 GRAI-96 Encoding Procedure**

1394 The following procedure creates a GRAI-96 encoding.

1395 Given:

- 1396 • A GS1 GRAI consisting of digits  $d_1d_2d_3\dots d_K$ , where  $14 \leq K \leq 25$ .
- 1397 • The length  $L$  of the Company Prefix portion of the GRAI
- 1398 • A Filter Value  $F$  where  $0 \leq F < 8$

1399 *Explanation (non-normative): Because a GRAI must include a serial number to be*  
1400 *convertible into an EPC, K must be at least 14 (that is, the serial number must contain at*  
1401 *least one character).*

1402 Procedure:

- 1403 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of  
1404 the Partition Table (Table 17) to determine the Partition Value,  $P$ , the number of bits  $M$  in  
1405 the Company Prefix field, and the number of bits  $N$  in Asset Type field. If  $L$  is not found in  
1406 any row of Table 17, stop: this GRAI cannot be encoded in a GRAI-96.
- 1407 2. Construct the Company Prefix by concatenating digits  $d_1d_2d_3\dots d_{(L)}$  and considering the  
1408 result to be a decimal integer,  $C$ .
- 1409 3. If  $L < 12$  construct the Asset Type by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and  
1410 considering the result to be a decimal integer,  $I$ . Otherwise set bits  $b_{41}, b_{40}, b_{39}, b_{38}$  to 0000.
- 1411 4. Construct the Serial Number by concatenating digits  $d_{14}d_{15}\dots d_K$ . If any of these  
1412 characters is not a digit, stop: this GRAI cannot be encoded in the GRAI-96 encoding.  
1413 Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^{38}$ , stop: this GRAI cannot  
1414 be encoded in the GRAI-96 encoding. Also, if  $K > 14$  and  $d_{14} = 0$ , stop: this GRAI cannot be  
1415 encoded in the GRAI-96 encoding (because leading zeros are not permitted except in the  
1416 case where the Serial Number consists of a single zero digit).
- 1417 5. Construct the final encoding by concatenating the following bit fields, from most  
1418 significant to least significant: Header 00110011 (8 bits), Filter Value  $F$  (3 bits), Partition  
1419 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Asset Type  $I$  from  
1420 Step 3 ( $N$  bits) and Serial Number  $S$  from Step 4 (38 bits). Note that  $M+N = 44$  bits for all  $P$ .

### 1421 **3.8.1.2 GRAI-96 Decoding Procedure**

1422 Given:

- 1423 • An GRAI-96 as a 96-bit bit string 00110011 $b_{87}b_{86}\dots b_0$  (where the first eight bits  
1424 00110011 are the header)

1425 Yields:

- 1426 • A GS1 GRAI

- 1427 • A Filter Value
- 1428 Procedure:
- 1429 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1430 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If
- 1431  $P = 7$ , stop: this bit string cannot be decoded as a GRAI-96.
- 1432 3. Look up the Partition Value  $P$  in Table 17 to obtain the number of bits  $M$  in the Company
- 1433 Prefix and the number of digits  $L$  in the Company Prefix.
- 1434 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80} \dots b_{(82-M)}$  as an unsigned integer.
- 1435 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal GRAI-96
- 1436 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2 \dots p_L$ , adding leading
- 1437 zeros as necessary to make up  $L$  digits in total.
- 1438 5. If  $L < 12$  extract the Asset Type by considering bits  $b_{(81-M)} b_{(80-M)} \dots b_{38}$  as an unsigned
- 1439 integer. If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit string is not a
- 1440 legal GRAI-96 encoding. Otherwise, convert this integer to a  $(12-L)$ -digit decimal number
- 1441  $i_1i_2 \dots i_{(12-L)}$ , adding leading zeros as necessary to make  $(12-L)$  digits.
- 1442 6. Construct a 12-digit number  $d_1d_2d_3 \dots d_{12}$  where  $d_1d_2 \dots d_{(L)} = p_1p_2 \dots p_L$  from Step 4, and
- 1443  $d_{(L+1)}d_{(L+2)} \dots d_{12} = i_1 i_2 \dots i_{(12-L)}$  from Step 5.
- 1444 7. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$
- 1445  $d_{11})) \bmod 10$ .
- 1446 8. Extract the Serial Number by considering bits  $b_{37}b_{36} \dots b_0$  as an unsigned integer. Convert
- 1447 this integer to a decimal number  $d_{14}d_{15} \dots d_K$ , with no leading zeros (exception: if the integer
- 1448 is equal to zero, convert it to a single zero digit).
- 1449 9. The GS1 GRAI is the concatenation of the digits from Steps 6, 7, and 8:  $d_1d_2d_3 \dots d_K$ .

1450 **3.8.2 GRAI-170**

1451 In addition to a Header, the GRAI-170 is composed of five fields: the *Filter Value*, *Partition*,

1452 *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

1453 \*Max. decimal value range of Company Prefix and Asset Type fields vary according to contents of the Partition  
1454 field.

1455 **Table 18.** The EPC GRAI-170 bit allocation, header, and maximum decimal values.

- 1456 • *Header* is 8-bits, with a binary value of 0011 0111
- 1457 • *Filter Value* is not part of the GRAI or EPC identifier, but is used for fast filtering and  
1458 pre-selection of basic asset types. The Filter Values for 96-bit and 170-bit GRAI are  
1459 shown in Table 16. Values marked as “reserved” are reserved for assignment by  
1460 EPCglobal in future versions of this specification. Implementations of the encoding  
1461 and decoding rules specified below SHALL accept any value of the filter bits, whether  
1462 reserved or not. Applications, however, SHOULD NOT direct an encoder to write a  
1463 reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so  
1464 may cause interoperability problems if a reserved value is assigned in a future revision  
1465 to this specification. This specification anticipates that valuable Filter Values will be  
1466 determined once there has been time to consider the possible use cases.
- 1467 • *Partition* is an indication of where the subsequent Company Prefix and Asset Type  
1468 numbers are divided. This organization matches the structure in the GS1 GRAI in  
1469 which the Company Prefix added to the Asset Type number totals 12 digits, yet the  
1470 Company Prefix may vary from 6 to 12 digits and the Asset Type from 6 to 0 digit(s).  
1471 The available values of *Partition* and the corresponding sizes of the *Company Prefix*  
1472 and *Asset Type* fields for 96-bit and 170-bit GRAI are defined in Table 17.
- 1473 • *Company Prefix* contains a literal embedding of the GS1 Company Prefix.
- 1474 • *Asset Type, if present*, encodes the GRAI Asset Type number.
- 1475 • *Serial Number* contains a mandatory alphanumeric serial number. The GRAI-170  
1476 encoding is capable of representing alphanumeric serial numbers of up to 16 characters,  
1477 permitting the full range of serial numbers available in the GS1-128 barcode  
1478 symbology using Application Identifier (AI) 8003 [GS1GS].

### 1479 3.8.2.1 GRAI-170 Encoding Procedure

1480 The following procedure creates a GRAI-170 encoding.

1481 Given:

- 1482 • A GS1 GRAI consisting of digits  $d_1d_2d_3\dots d_{13}$ , and a variable length alphanumeric serial  
1483 number  $s_{14}s_{15}\dots s_K$  where  $14 \leq K \leq 29$ .
- 1484 • The length  $L$  of the Company Prefix portion of the GRAI
- 1485 • A Filter Value  $F$  where  $0 \leq F < 8$

1486 *Explanation (non-normative): Because a GRAI must include a serial number to be*  
1487 *convertible into an EPC, K must be at least 14 (that is, the serial number must contain at*  
1488 *least one character).*

1489

1490 Procedure:

1491 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of  
 1492 the Partition Table (Table 17) to determine the Partition Value,  $P$ , the number of bits  $M$  in  
 1493 the Company Prefix field, and the number of bits  $N$  in Asset Type field. If  $L$  is not found in  
 1494 any row of Table 17, stop: this GRAI cannot be encoded in a GRAI-170.

1495 2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_{(L)}$  and considering the  
 1496 result to be a decimal integer,  $C$ .

1497 3. If  $L < 12$  construct the Asset Type by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and  
 1498 considering the result to be a decimal integer,  $I$ . Otherwise set bits  $b_{115}, b_{114}, b_{113}, b_{112}$  to 0000.

1499 4. Check that each of the characters  $s_{14}s_{15}\dots s_K$  is one of the 82 characters listed in the table  
 1500 in Appendix F. If this is not the case, stop: this character string cannot be encoded as a  
 1501 GRAI-170. Otherwise construct the Serial Number by concatenating the 7-bit code, as given  
 1502 in Appendix F, for each of the characters  $s_{14}s_{15}\dots s_K$ , yielding  $7*(K-14)$  bits total. If  $K < 29$ ,  
 1503 concatenate additional zero bits to the right to make a total of 112 bits.

1504 5. Construct the final encoding by concatenating the following bit fields, from most  
 1505 significant to least significant: Header 00110111 (8 bits), Filter Value  $F$  (3 bits), Partition  
 1506 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Asset Type  $I$  from  
 1507 Step 3 ( $N$  bits) and Serial Number  $S$  from Step 4 (112 bits). Note that  $M+N = 44$  bits for  
 1508 all  $P$ .

### 1509 3.8.2.2 GRAI-170 Decoding Procedure

1510 Given:

1511 • An GRAI-170 as a 170-bit bit string  $00110111b_{161}b_{160}\dots b_0$  (where the first eight bits  
 1512  $00110111$  are the header)

1513 Yields:

1514 • A GS1 GRAI

1515 • A Filter Value

1516 Procedure:

1517 1. Bits  $b_{161}b_{160}b_{159}$ , considered as an unsigned integer, are the Filter Value.

1518 2. Extract the Partition Value  $P$  by considering bits  $b_{158}b_{157}b_{156}$  as an unsigned integer. If  
 1519  $P = 7$ , stop: this bit string cannot be decoded as a GRAI-170.

1520 3. Look up the Partition Value  $P$  in Table 17 to obtain the number of bits  $M$  in the Company  
 1521 Prefix and the number of digits  $L$  in the Company Prefix.

1522 4. Extract the Company Prefix  $C$  by considering bits  $b_{155}b_{154}\dots b_{(156-M)}$  as an unsigned  
 1523 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal  
 1524 GRAI-170 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ ,  
 1525 adding leading zeros as necessary to make up  $L$  digits in total.

1526 5. If  $L < 12$  extract the Asset Type by considering bits  $b_{(155-M)} b_{(154-M)}\dots b_{112}$  as an unsigned  
 1527 integer. If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit string is not a

1528 legal GRAI-170 encoding. Otherwise, convert this integer to a (12-L)-digit decimal number  
 1529  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as necessary to make (12-L) digits.

1530 6. Construct a 12-digit number  $d_1d_2d_3\dots d_{12}$  where  $d_1d_2\dots d_{(L)} = p_1p_2\dots p_L$  from Step 4, and  
 1531  $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_1 i_2\dots i_{(12-L)}$  from Step 5.

1532 7. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$   
 1533  $d_{11})) \bmod 10$ .

1534 8. Divide the remaining bits  $b_{111}b_{110}\dots b_0$  into 7-bit segments. This string should consist of  
 1535 K non-zero segments followed by 16-K zero segments. If this is not the case, stop: this bit  
 1536 string cannot be decoded as a GRAI-170. Otherwise, look up each of the non-zero 7-bit  
 1537 segments in Appendix F to obtain a corresponding character. If any of the non-zero 7-bit  
 1538 segments has a value that is not in Appendix F, stop: this bit string cannot be decoded as a  
 1539 GRAI-170. Otherwise, the first K characters considered as a character string is the serial  
 1540 number  $s_{14}s_{15}\dots s_K$ .

1541 9. The GS1 GRAI is the concatenation of the digits from Steps 6 and 7 and the characters  
 1542 from Step 8. :  $d_1d_2\dots d_{13} s_{14}s_{15}\dots s_K$

1543

1544 **3.9 Global Individual Asset Identifier (GIAI)**

1545 The EPC Tag Encoding scheme for GIAI permits the direct embedding of GS1 System  
 1546 standard GIAI codes on EPC tags.

1547 **3.9.1 GIAI-96**

1548 In addition to a Header, the EPC GIAI-96 is composed of four fields: the *Filter Value*,  
 1549 *Partition*, *Company Prefix*, and *Individual Asset Reference*, as shown in Table 19.

1550

	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*)

1551

1552 \*Max. decimal value range of Company Prefix and Individual Asset Reference fields vary according to contents  
 1553 of the Partition field.

1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564

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

- *Header* is 8-bits, with a binary value of 0011 0100.
- *Filter Value* is not part of the GIAI or EPC identifier, but is used for fast filtering and pre-selection of basic asset types. The Filter Values for 96-bit and 202-bit GIAI are the same shown in Table 20. Values marked as “reserved” are reserved for assignment by EPCglobal in future versions of this specification. Implementations of the encoding and decoding rules specified below SHALL accept any value of the filter bits, whether reserved or not. Applications, however, SHOULD NOT direct an encoder to write a reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so may cause interoperability problems if a reserved value is assigned in a future revision to this specification.

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

1565

**Table 20.** GIAI Filter Values

1566  
1567  
1568  
1569  
1570

- The *Partition* is an indication of where the subsequent Company Prefix and Individual Asset Reference numbers are divided. This organization matches the structure in the GS1 GIAI in which the Company Prefix may vary from 6 to 12 digits. The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *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	13
1	37	11	45	14
2	34	10	48	15
3	30	9	52	16

Partition Value ( <i>P</i> )	Company Prefix		Individual Asset Reference	
	Bits ( <i>M</i> )	Digits ( <i>L</i> )	Bits ( <i>N</i> )	Digits
4	27	8	55	17
5	24	7	58	18
6	20	6	62	19

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

1571

1572

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

1573

- *Individual Asset Reference* is a mandatory unique number for each instance. The EPC representation is only capable of representing a subset of asset references allowed in the GS1 General Specifications. The capacity of this asset reference is less than the maximum GS1 System specification for asset references, no leading zeros are permitted, and only numbers are permitted.

1574

1575

1576

1577

1578

### 3.9.1.1 GIAI-96 Encoding Procedure

1579

The following procedure creates a GIAI-96 encoding.

1580

Given:

1581

- A GS1 GIAI consisting of digits  $d_1d_2\dots d_K$ , where  $K \leq 30$ .

1582

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

1583

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

1584

Procedure:

1585

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

1586

1587

1588

1589

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

1590

1591

3. Construct the Individual Asset Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_K$ . If any of these characters is not a digit, stop: this GIAI cannot be encoded in the GIAI-96 encoding. Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^N$ , stop: this GIAI cannot be encoded in the GIAI-96 encoding. Also, if  $K > L+1$  and  $d_{(L+1)} = 0$ , stop: this GIAI cannot be encoded in the GIAI-96 encoding (because leading zeros are not permitted except in the case where the Individual Asset Reference consists of a single zero digit).

1592

1593

1594

1595

1596

1597

4. Construct the final encoding by concatenating the following bit fields, from most significant to least significant: Header 00110100 (8 bits), Filter Value  $F$  (3 bits), Partition

1598

1599 Value  $P$  from Step 2 (3 bits), Company Prefix  $C$  from Step 3 ( $M$  bits) and Individual Asset  
1600 Number  $S$  from Step 4 ( $N$  bits). Note that  $M+N = 82$  bits for all  $P$ .

### 1601 **3.9.1.2 GIAI-96 Decoding Procedure**

1602 Given:

- 1603 • A GIAI-96 as a 96-bit bit string  $00110100b_{87}b_{86}\dots b_0$  (where the first eight bits  
1604  $00110100$  are the header)

1605 Yields:

- 1606 • A GS1 GIAI
- 1607 • A Filter Value

1608 Procedure:

- 1609 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1610 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
1611  $P = 7$ , stop: this bit string cannot be decoded as a GIAI-96.
- 1612 3. Look up the Partition Value  $P$  in Table 21 to obtain the number of bits  $M$  in the Company  
1613 Prefix and the number of digits  $L$  in the Company Prefix.
- 1614 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned integer.  
1615 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal GIAI-96  
1616 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ , adding leading  
1617 zeros as necessary to make up  $L$  digits in total.
- 1618 5. Extract the Individual Asset Reference by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_0$  as an  
1619 unsigned integer. If this integer is greater than or equal to  $10^{(30-L)}$ , stop: the input bit string  
1620 is not a legal GIAI-96 encoding. Otherwise, convert this integer to a decimal number  
1621  $s_1s_2\dots s_J$ , with no leading zeros (exception: if the integer is equal to zero, convert it to a single  
1622 zero digit).
- 1623 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  
1624  $d_{(L+1)}d_{(L+2)}\dots d_K = s_1s_2\dots s_J$  from Step 5. This  $K$ -digit number, where  $K \leq 30$ , is the GS1 GIAI.

### 1625 **3.9.2 GIAI-202**

1626 In addition to a Header, the EPC GIAI-202 is composed of four fields: the *Filter Value*,  
1627 *Partition*, *Company Prefix*, and *Individual Asset Reference*, as shown in Table 22.

1628

	Header	Filter Value	Partition	Company Prefix	Individual Asset Reference
GIAI-202	8	3	3	20-40	168-148
	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

1629

1630  
1631

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

1632

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

1633

- *Header* is 8-bits, with a binary value of 0011 1000.

1634

- *Filter Value* is not part of the GIAI or EPC identifier, but is used for fast filtering and pre-selection of basic asset types. The Filter Values for 96-bit and 202-bit GIAI are the same shown in Table 20. Values marked as “reserved” are reserved for assignment by EPCglobal in future versions of this specification. Implementations of the encoding and decoding rules specified below SHALL accept any value of the filter bits, whether reserved or not. Applications, however, SHOULD NOT direct an encoder to write a reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so may cause interoperability problems if a reserved value is assigned in a future revision to this specification.

1635

1636

1637

1638

1639

1640

1641

1642

1643

- The *Partition* is an indication of the size of the subsequent Company Prefix. This organization matches the structure in the GS1 GIAI in which the Company Prefix may vary from 6 to 12 digits. The available values of *Partition* and the corresponding size of the *Company Prefix* field is defined in Table 23.

1644

1645

1646

1647

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

1648

1649

**Table 23.** GIAI-202 Partitions.

1650

- *Company Prefix* contains a literal embedding of the GS1 Company Prefix.

1651

- *Individual Asset Reference* contains a mandatory alphanumeric asset reference number. The GIAI-202 encoding is capable of representing alphanumeric serial numbers of up to 24 characters, permitting the full range of serial numbers available in the GS1-128 barcode symbology using Application Identifier (AI) 8004 [GS1GS].

1652

1653

1654

1655

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

1656

1657

### 3.9.2.1 GIAI-202 Encoding Procedure

1658

1659

The following procedure creates a GIAI-202 encoding.

1660

Given:

1661

- A GS1 GIAI consisting of digits  $d_1d_2d_3\dots d_L$ , and a variable length alphanumeric serial number  $s_{L+1}s_{L+2}\dots s_K$  where  $L+1 \leq K \leq 30$ .

1662

1663

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

1664

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

1665

Procedure:

1666

1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of the Partition Table (Table 23) to determine the Partition Value,  $P$ , the number of bits  $M$  in the Company Prefix field, and the number of bits  $N$  in the Individual Asset Reference field. If  $L$  is not found in any row of Table 23, stop: this GIAI cannot be encoded in a GIAI-202.

1667

1668

1669

1670

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

1671

1672

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

1673

1674 GIAI-202. Otherwise construct the Individual Asset Reference by concatenating the 7-bit  
1675 code, as given in Appendix F, for each of the characters  $s_{(L+1)}s_{(L+2)}\dots s_K$  yielding  $7*(K-L)$   
1676 bits total. Concatenate additional zero bits to the right, if necessary, to make a total of  $(188-$   
1677  $M)$  bits, where  $M$  is the number of bits in the Company Prefix portion as determined in Step  
1678 1.

1679 4. Construct the final encoding by concatenating the following bit fields, from most  
1680 significant to least significant: Header 00111000 (8 bits), Filter Value  $F$  (3 bits), Partition  
1681 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits) and Individual Asset  
1682 Number  $S$  from Step 3 ( $188-M$  bits).

1683

### 1684 **3.9.2.2 GIAI-202 Decoding Procedure**

1685 Given:

- 1686 • A GIAI-202 as a 202-bit bit string  $00111000b_{193}b_{192}\dots b_0$  (where the first eight bits  
1687  $00111000$  are the header)

1688 Yields:

- 1689 • A GS1 GIAI
- 1690 • A Filter Value

1691 Procedure:

1692 1. Bits  $b_{193}b_{192}b_{191}$ , considered as an unsigned integer, are the Filter Value.

1693 2. Extract the Partition Value  $P$  by considering bits  $b_{190}b_{189}b_{188}$  as an unsigned integer. If  
1694  $P = 7$ , stop: this bit string cannot be decoded as a GIAI-202.

1695 3. Look up the Partition Value  $P$  in Table 23 to obtain the number of bits  $M$  in the Company  
1696 Prefix and the number of digits  $L$  in the Company Prefix.

1697 4. Extract the Company Prefix  $C$  by considering bits  $b_{187}b_{186}\dots b_{(188-M)}$  as an unsigned  
1698 integer. If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal  
1699 GIAI-202 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ , adding  
1700 leading zeros as necessary to make up  $L$  digits in total.

1701 5. Extract the Individual Asset Reference by dividing the remaining bits  $b_{(187-M)}b_{(186-M)}\dots b_0$   
1702 into 7 bit segments beginning with the segment  $b_{(187-M)}b_{(186-M)}\dots b_{(181-M)}$ , and continuing as  
1703 far as possible (there may be up to four bits left over at the end).. The result should consist  
1704 of  $J$  non-zero segments followed by zero or more zero-valued segments, with any remaining  
1705 bits also being zeros. If this is not the case, stop: this bit string cannot be decoded as a GIAI  
1706 -202. Otherwise, look up each of the non-zero 7-bit segments in Appendix F to obtain a  
1707 corresponding character. If any of the non-zero 7-bit segments has a value that is not in  
1708 Appendix F, stop: this bit string cannot be decoded as a GIAI-202. Otherwise, the first  $J$   
1709 characters considered as a character string is the Asset Reference Number  $s_{(1)}s_{(2)}\dots s_J$ .

1710 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  
1711  $s_{(L+1)}s_{(L+2)}\dots s_K = s_{(1)}s_{(2)}\dots s_J$  from Step 5. This  $K$ -character string, where  $K \leq 30$ , is the GS1  
1712 GIAI.

1713

1714 **3.10 Global Service Relation Number (GSRN)**

1715 The EPC Tag Encoding scheme for GSRN permits the direct embedding of GS1 System  
1716 standard GSRN codes on EPC tags. In all cases, the check digit is not encoded.

1717 **3.10.1 GSRN-96**

1718 In addition to a Header, the EPC GSRN-96 is composed of four fields: the *Filter Value*,  
1719 *Partition*, *Company Prefix*, and *Service Reference*, as shown in Table 24.

1720

	Header	Filter Value	Partition	Company Prefix	Service Reference	Unallocated
GSRN-96	8	3	3	20-40	38-18	24
	0010 1101 (Binary value)	(Refer to Table 25 for values )	(Refer to Table 26 for values )	999,999 – 999,999,999,999 (Max. decimal range*)	99,999,999,999 – 99,999 (Max. decimal range*)	[Not Used]

1721

1722 \*Max. decimal value range of Company Prefix and Service Reference fields vary according to the contents of the  
1723 Partition field.

1724 **Table 24.** The EPC 96-bit GSRN bit allocation, header, and maximum decimal values.

- 1725 • *Header* is 8-bits, with a binary value of 0010 1101.
- 1726 • *Filter Value* is not part of the GSRN or EPC identifier, but is used for fast filtering and  
1727 pre-selection of basic document types. The normative specifications for GSRN Filter  
1728 Values are specified in Table 25. Values marked as “reserved” are reserved for  
1729 assignment by EPCglobal in future versions of this specification. Implementations of  
1730 the encoding and decoding rules specified below SHALL accept any value of the filter  
1731 bits, whether reserved or not. Applications, however, SHOULD NOT direct an  
1732 encoder to write a reserved value to a tag, nor rely upon a reserved value decoded from  
1733 a tag, as doing so may cause interoperability problems if a reserved value is assigned in  
1734 a future revision to this specification.

1735

Type	Binary Value
All Others	000
Reserved	001

Type	Binary Value
Reserved	010
Reserved	011
Reserved	100
Reserved	101
Reserved	110
Reserved	111

**Table 25.** GSRN Filter Values

1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743

- The *Partition* is an indication of where the subsequent Company Prefix and Serial Reference numbers are divided. This organization matches the structure in the GS1 GSRN in which the Company Prefix added to the Service Reference number totals 17 digits, yet the Company Prefix may vary from 6 to 12 digits and the Service Reference from 11 to 5 digits. Table 26 shows allowed values of the partition value and the corresponding lengths of the company prefix and service reference.

Partition Value ( <i>P</i> )	Company Prefix		Service 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 26.** GSRN-96 Partitions.

1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751

- *Company Prefix* contains a literal embedding of the Company Prefix.
- *Service Reference*, a unique number for each instance, is treated as a single integer, and encoded into binary to form the Service Reference field. The Service Reference must not exceed the capacity specified in GS1 specifications, which are 99,999 for company prefixes of 12 digits up to 99,999,999,999 for company prefixes of 6 digits.
- *Unallocated* is not used. This field must contain zeros to conform to this version of the specification.

1752 **3.10.1.1 GSRN-96 Encoding Procedure**

1753 The following procedure creates a GSRN-96 encoding.

1754 Given:

- 1755 • A GS1 GSRN consisting of digits  $d_1d_2\dots d_{18}$
- 1756 • The length  $L$  of the Company Prefix portion of the GSRN
- 1757 • A Filter Value  $F$  where  $0 \leq F < 8$

1758 Procedure:

- 1759 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of  
1760 the Partition Table (Table 26) to determine the Partition Value,  $P$ , the number of bits  $M$  in  
1761 the Company Prefix field, and the number of bits  $N$  in the Service Reference. If  $L$  is not  
1762 found in any row of Table 26, stop: this GSRN cannot be encoded in a GSRN-96.
- 1763 2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_{(L)}$  and considering the  
1764 result to be a decimal integer,  $C$ .
- 1765 3. Construct the Service Reference by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{17}$  and considering  
1766 the result to be a decimal integer,  $S$ .
- 1767 4. Construct the final encoding by concatenating the following bit fields, from most  
1768 significant to least significant: Header 00101101 (8 bits), Filter Value  $F$  (3 bits), Partition  
1769 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Service Reference  $S$   
1770 from Step 3 ( $N$  bits), and 24 zero bits. Note that  $M+N = 58$  bits for all  $P$ .

1771 **3.10.1.2 GSRN-96 Decoding Procedure**

1772 Given:

- 1773 • A GSRN-96 as a 96-bit bit string  $00101101b_{87}b_{86}\dots b_0$  (where the first eight bits  
1774  $00101101$  are the header)

1775 Yields:

- 1776 • A GS1 GSRN
- 1777 • A Filter Value

1778 Procedure:

- 1779 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1780 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
1781  $P = 7$ , stop: this bit string cannot be decoded as a GSRN-96.
- 1782 3. Look up the Partition Value  $P$  in Table 10 to obtain the number of bits  $M$  in the Company  
1783 Prefix and the number of digits  $L$  in the Company Prefix.
- 1784 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned integer.  
1785 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal GSRN-96  
1786 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ , adding leading  
1787 zeros as necessary to make up  $L$  digits in total.

- 1788 5. Extract the Service Reference by considering bits  $b_{(81-M)} b_{(80-M)} \dots b_{24}$  as an unsigned  
 1789 integer. If this integer is greater than or equal to  $10^{(17-L)}$ , stop: the input bit string is not a  
 1790 legal GSRN-96 encoding. Otherwise, convert this integer to a (17-L)-digit decimal number  
 1791  $i_1 i_2 \dots i_{(17-L)}$ , adding leading zeros as necessary to make (17-L) digits.
- 1792 6. Construct a 17-digit number  $d_1 d_2 \dots d_{17}$  where  $d_1 d_2 \dots d_{(L)} = p_1 p_2 \dots p_L$  from Step 4, and  
 1793  $d_{(L+1)} d_{(L+2)} \dots d_{17} = i_1 i_2 \dots i_{(17-L)}$  from Step 5.
- 1794 7. Calculate the check digit  $d_{18} = (-3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11} + d_{13} + d_{15} + d_{17}) - (d_2 + d_4$   
 1795  $+ d_6 + d_8 + d_{10} + d_{12} + d_{14} + d_{16})) \bmod 10$ .
- 1796 8. The GS1 GSRN is the concatenation of digits from Steps 6 and 7:  $d_1 d_2 \dots d_{18}$ .

1797 **3.11 Global Document Type Identifier (GDTI)**

1798 The EPC Tag Encoding scheme for GDTI permits the direct embedding of GS1 System  
 1799 standard GDTI on EPC tags. In all cases, the check digit is not encoded. Only GDTIs that  
 1800 include the optional serial number may be represented as EPCs. A GDTI without a serial  
 1801 number represents an document class, rather than a specific instance, and therefore may not  
 1802 be used as an EPC (just as a non-serialized GTIN may not be used as an EPC).

1803 **3.11.1 GDTI-96**

1804 In addition to a Header, the GDTI-96 is composed of five fields: the *Filter Value*, *Partition*,  
 1805 *Company Prefix*, *Document Type*, and *Serial Number*, as shown in Table 27.

	Header	Filter Value	Partition	Company Prefix	Document Type	Serial Number
GDTI-96	8	3	3	20-40	21-1	41
	0010 1100 (Binary value)	(Refer to Table 28 for values)	(Refer to Table 29 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)

1806 \*Max. decimal value range of Company Prefix and Asset Type fields vary according to contents of the Partition  
 1807 field.

1808 **Table 27.** The EPC GDTI-96 bit allocation, header, and maximum decimal values.

- 1809 • *Header* is 8-bits, with a binary value of 0010 1100.
- 1810 • *Filter Value* is not part of the GDTI or EPC identifier, but is used for fast filtering and  
 1811 pre-selection of basic document types. The Filter Values for 96-bit and 113-bit GDTI  
 1812 are the same shown in Table 28. Values marked as “reserved” are reserved for  
 1813 assignment by EPCglobal in future versions of this specification. Implementations of

1814  
1815  
1816  
1817  
1818

the encoding and decoding rules specified below SHALL accept any value of the filter bits, whether reserved or not. Applications, however, SHOULD NOT direct an encoder to write a reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so may cause interoperability problems if a reserved value is assigned in a future revision to this specification.

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

**Table 28.** GDTI Filter Values

1819  
1820  
1821  
1822  
1823  
1824  
1825

- *Partition* is an indication of where the subsequent Company Prefix and Document Type numbers are divided. This organization matches the structure in the GS1 GDTI in which the Company Prefix added to the Document Type number totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the Document Type from 6 to 0 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Document Type* fields are defined in Table 29.

Partition Value ( <i>P</i> )	Company Prefix		Document Type	
	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 29.** GDTI Partitions.

1826

1827

- 1828 • *Company Prefix* contains a literal embedding of the GS1 Company Prefix.
- 1829 • *Document Type*, if present, *encodes* the GDTI Document Type number.
- 1830 • *Serial Number* contains a numeric serial number. The 96-bit tag encodings are only
- 1831 capable of representing a subset of Serial Numbers allowed in the GS1 General
- 1832 Specifications. The capacity of this numeric serial number is less than the maximum
- 1833 GS1 System specification for this serial number and no leading zeros are permitted.

### 1834 3.11.1.1 GDTI-96 Encoding Procedure

1835 The following procedure creates a GDTI-96 encoding.

1836 Given:

- 1837 • A GS1 GDTI consisting of digits  $d_1d_2\dots d_K$ , where  $14 \leq K \leq 26$ .
- 1838 • The length  $L$  of the Company Prefix portion of the GDTI
- 1839 • A Filter Value  $F$  where  $0 \leq F < 8$

1840 *Explanation (non-normative): Because a GDTI must include a serial number to be*  
1841 *convertible into an EPC, K must be at least 14 (that is, the serial number must contain at*  
1842 *least one character).*

1843

1844 Procedure:

- 1845 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of
- 1846 the Partition Table (Table 17) to determine the Partition Value,  $P$ , the number of bits  $M$  in
- 1847 the Company Prefix field, and the number of bits  $N$  in Document Type field. If  $L$  is not
- 1848 found in any row of Table 17, stop: this GDTI cannot be encoded in a GDTI-96.
- 1849 2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_{(L)}$  and considering the
- 1850 result to be a decimal integer,  $C$ .
- 1851 3. If  $L < 12$  construct the Document Type by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and
- 1852 considering the result to be a decimal integer,  $I$ . If  $L = 12$  set bit  $b_{41}$  to 0 since there is no
- 1853 Document Type digit.
- 1854 4. Construct the Serial Number by concatenating digits  $d_{14}d_{15}\dots d_K$ . If any of these
- 1855 characters is not a digit, stop: this GDTI cannot be encoded in the GDTI-96 encoding.
- 1856 Otherwise, consider the result to be a decimal integer,  $S$ . If  $S \geq 2^{41}$ , stop: this GDTI cannot
- 1857 be encoded in the GDTI-96 encoding. Also, if  $K > 14$  and  $d_{14} = 0$ , stop: this GDTI cannot be
- 1858 encoded in the GDTI-96 encoding (because leading zeros are not permitted except in the
- 1859 case where the Serial Number consists of a single zero digit).
- 1860 5. Construct the final encoding by concatenating the following bit fields, from most
- 1861 significant to least significant: Header 00101100 (8 bits), Filter Value  $F$  (3 bits), Partition
- 1862 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Document Type  $I$
- 1863 from Step 3 ( $N$  bits) and Serial Number  $S$  from Step 4 (41 bits). Note that  $M+N = 41$  bits for
- 1864 all  $P$ .

1865 **3.11.1.2 GDTI-96 Decoding Procedure**

1866 Given:

- 1867 • A GDTI-96 as a 96-bit bit string  $00101100b_{87}b_{86}\dots b_0$  (where the first eight bits  
1868  $00101100$  are the header)

1869 Yields:

- 1870 • A GS1 GDTI  
1871 • A Filter Value

1872 Procedure:

- 1873 1. Bits  $b_{87}b_{86}b_{85}$ , considered as an unsigned integer, are the Filter Value.
- 1874 2. Extract the Partition Value  $P$  by considering bits  $b_{84}b_{83}b_{82}$  as an unsigned integer. If  
1875  $P = 7$ , stop: this bit string cannot be decoded as a GDTI-96.
- 1876 3. Look up the Partition Value  $P$  in Table 17 to obtain the number of bits  $M$  in the Company  
1877 Prefix and the number of digits  $L$  in the Company Prefix.
- 1878 4. Extract the Company Prefix  $C$  by considering bits  $b_{81}b_{80}\dots b_{(82-M)}$  as an unsigned integer.  
1879 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal GDTI-96  
1880 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2\dots p_L$ , adding leading  
1881 zeros as necessary to make up  $L$  digits in total.
- 1882 5. If  $L < 12$  extract the Document Type by considering bits  $b_{(81-M)}b_{(80-M)}\dots b_{41}$  as an  
1883 unsigned integer. If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit string  
1884 is not a legal GDTI-96 encoding. Otherwise, convert this integer to a  $(12-L)$ -digit decimal  
1885 number  $i_1i_2\dots i_{(12-L)}$ , adding leading zeros as necessary to make  $(12-L)$  digits.
- 1886 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  
1887  $d_{(L+1)}d_{(L+2)}\dots d_{12} = i_1i_2\dots i_{(12-L)}$  from Step 5.
- 1888 7. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$   
1889  $d_{11})) \bmod 10$ .
- 1890 8. Extract the Serial Number by considering bits  $b_{40}b_{39}\dots b_0$  as an unsigned integer. Convert  
1891 this integer to a decimal number  $d_{14}d_{15}\dots d_K$ , with no leading zeros (exception: if the integer  
1892 is equal to zero, convert it to a single zero digit).
- 1893 9. The GS1 GDTI is the concatenation of the digits from Steps 6, 7, and 8:  $d_1d_2\dots d_K$ .

1894 **3.11.2 GDTI-113**

1895 In addition to a Header, the GDTI-113 is composed of five fields: the *Filter Value*, *Partition*,  
1896 *Company Prefix*, *Asset Type*, and *Serial Number*, as shown in Table 30.

	Header	Filter Value	Partition	Company Prefix	Document Type	Serial Number
GDTI-113	8	3	3	20-40	21-1	58
	0011 1010 (Binary value)	(Refer to Table 28 for values )	(Refer to Table 29 for values )	999,999 – 999,999,999,999 (Max. decimal range*)	999,999 – 0 (Max. decimal range*)	Up to 17 numeric characters

1897 \*Max. decimal value range of Company Prefix and Asset Type fields vary according to contents of the Partition  
1898 field.

1899 **Table 30.** The EPC GDTI-113 bit allocation, header, and maximum decimal values.

- 1900 • *Header* is 8-bits, with a binary value of 00111010
- 1901 • *Filter Value* is not part of the GDTI or EPC identifier, but is used for fast filtering and  
1902 pre-selection of basic asset types. The Filter Values for 96-bit and 113-bit GDTI are  
1903 the same shown in Table 28. Values marked as “reserved” are reserved for assignment  
1904 by EPCglobal in future versions of this specification. Implementations of the encoding  
1905 and decoding rules specified below SHALL accept any value of the filter bits, whether  
1906 reserved or not. Applications, however, SHOULD NOT direct an encoder to write a  
1907 reserved value to a tag, nor rely upon a reserved value decoded from a tag, as doing so  
1908 may cause interoperability problems if a reserved value is assigned in a future revision  
1909 to this specification. This specification anticipates that valuable Filter Values will be  
1910 determined once there has been time to consider the possible use cases.
- 1911 • *Partition* is an indication of where the subsequent Company Prefix and Document Type  
1912 numbers are divided. This organization matches the structure in the GS1 GDTI in  
1913 which the Company Prefix added to the Document Type number totals 12 digits, yet  
1914 the Company Prefix may vary from 6 to 12 digits and the Document Type from 6 to 0  
1915 digit(s). The available values of *Partition* and the corresponding sizes of the *Company*  
1916 *Prefix* and *Document Type* fields for 96-bit and 113-bit GDTI are defined in Table 29.
- 1917 • *Company Prefix* contains a literal embedding of the GS1 Company Prefix.
- 1918 • *Document Type, if present*, encodes the GDTI Document Type number.
- 1919 • *Serial Number* contains a numeric serial number. The GDTI-113 encoding is capable of  
1920 representing numeric serial numbers of up to 17 numeric characters including leading  
1921 zeros, permitting the full range of serial numbers specified in GS1 Standards using  
1922 Application Identifier (AI) 253 [GS1GS].

### 1923 3.11.2.1 GDTI-113 Encoding Procedure

1924 The following procedure creates a GDTI-113 encoding.

1925 Given:

- 1926 • A GS1 GDTI consisting of digits  $d_1d_2\dots d_{13}$ , and a variable length numeric serial number  
1927  $s_{14}s_{15}\dots s_K$  where  $14 \leq K \leq 30$ .
- 1928 • The length  $L$  of the Company Prefix portion of the GDTI
- 1929 • A Filter Value  $F$  where  $0 \leq F < 8$

1930 *Explanation (non-normative): Because a GDTI must include a serial number to be*  
1931 *convertible into an EPC,  $K$  must be at least 14 (that is, the serial number must contain at*  
1932 *least one character).*

1933 Procedure:

- 1934 1. Look up the length  $L$  of the Company Prefix in the “Company Prefix Digits” column of  
1935 the Partition Table (Table 17) to determine the Partition Value,  $P$ , the number of bits  $M$  in  
1936 the Company Prefix field, and the number of bits  $N$  in Document Type field. If  $L$  is not  
1937 found in any row of Table 17, stop: this GDTI cannot be encoded in a GDTI-113.
- 1938 2. Construct the Company Prefix by concatenating digits  $d_1d_2\dots d_{(L)}$  and considering the  
1939 result to be a decimal integer,  $C$ .
- 1940 3. If  $L < 12$  construct the Document Type by concatenating digits  $d_{(L+1)}d_{(L+2)}\dots d_{12}$  and  
1941 considering the result to be a decimal integer,  $I$ . If  $L = 12$  set bit  $b_{58}$  to 0 since there is no  
1942 Document Type digit.
- 1943 4. Construct the Serial Number by concatenating the digit 1 with digits  $d_{14}d_{15}\dots d_K$ . If any of  
1944 these characters is not a digit, stop: this GDTI cannot be encoded in the GDTI-113 encoding.  
1945 Otherwise, consider the result to be a decimal integer,  $S$ .
- 1946 5. Construct the final encoding by concatenating the following bit fields, from most  
1947 significant to least significant: Header 00111010 (8 bits), Filter Value  $F$  (3 bits), Partition  
1948 Value  $P$  from Step 1 (3 bits), Company Prefix  $C$  from Step 2 ( $M$  bits), Document Type  $I$   
1949 from Step 3 ( $N$  bits) and Serial Number  $S$  from Step 4 (58 bits). Note that  $M+N = 41$  bits for  
1950 all  $P$ .
- 1951

### 1952 3.11.2.2 GDTI-113 Decoding Procedure

1953 Given:

- 1954 • A GDTI-113 as a 113-bit bit string  $00111010b_{104}b_{103}\dots b_0$  (where the first eight bits  
1955  $00111010$  are the header)

1956 Yields:

- 1957 • A GS1 GDTI
- 1958 • A Filter Value

1959 Procedure:

- 1960 1. Bits  $b_{104}b_{103}b_{102}$ , considered as an unsigned integer, are the Filter Value.

- 1961 2. Extract the Partition Value  $P$  by considering bits  $b_{101}b_{100}b_{99}$  as an unsigned integer. If  
1962  $P = 7$ , stop: this bit string cannot be decoded as a GDTI-113.
- 1963 3. Look up the Partition Value  $P$  in Table 17 to obtain the number of bits  $M$  in the Company  
1964 Prefix and the number of digits  $L$  in the Company Prefix.
- 1965 4. Extract the Company Prefix  $C$  by considering bits  $b_{98}b_{97}...b_{(99-M)}$  as an unsigned integer.  
1966 If this integer is greater than or equal to  $10^L$ , stop: the input bit string is not a legal GDTI-  
1967 113 encoding. Otherwise, convert this integer into a decimal number  $p_1p_2...p_L$ , adding  
1968 leading zeros as necessary to make up  $L$  digits in total.
- 1969 5. If  $L < 12$  extract the Document Type by considering bits  $b_{(98-M)}b_{(97-M)}...b_{58}$  as an  
1970 unsigned integer. If this integer is greater than or equal to  $10^{(12-L)}$ , stop: the input bit string  
1971 is not a legal GDTI-113 encoding. Otherwise, convert this integer to a  $(12-L)$ -digit decimal  
1972 number  $i_1i_2...i_{(12-L)}$ , adding leading zeros as necessary to make  $(12-L)$  digits.
- 1973 6. . Construct a 12-digit number  $d_1d_2...d_{12}$  where  $d_1d_2...d_{(L)} = p_1p_2...p_L$  from Step 4, and  
1974  $d_{(L+1)}d_{(L+2)}...d_{12} = i_1i_2...i_{(12-L)}$  from Step 5.
- 1975 7. Calculate the check digit  $d_{13} = (-3(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}) - (d_1 + d_3 + d_5 + d_7 + d_9 +$   
1976  $d_{11})) \bmod 10$ .
- 1977 8. Extract the Serial Number by considering bits  $b_{57}b_{56}...b_0$  as an unsigned integer. Convert  
1978 this integer to a decimal number  $d_xd_{14}d_{15}...d_K$ , adding no leading zeros. If the first digit  $d_x$  is  
1979 not equal to 1, stop: the input text string is not a legal GDTI-113. Otherwise, remove the  
1980 leading 1 digit leaving  $d_{14}d_{15}...d_K$ .
- 1981 9. The GS1 GDTI is the concatenation of the digits from Steps 6, 7, and 8:  $d_1d_2...d_K$ .
- 1982
- 1983
- 1984

## 1985 3.12 DoD Tag Data Constructs

### 1986 3.12.1 DoD-96

1987 This tag data construct may be used to encode Class 1 tags for shipping goods to the United  
1988 States Department of Defense by an entity who has already been assigned a CAGE  
1989 (Commercial and Government Entity) code.

1990 At the time of this writing, the details of what information to encode into these fields is  
1991 explained in a document titled "United States Department of Defense Supplier's Passive  
1992 RFID Information Guide" that can be obtained at the United States Department of Defense's  
1993 website <http://www.dodrfid.org/suppliernguide.htm> .

1994 The current encoding structure of DoD-96 Tag Data Construct is shown in Table 31 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)

**Table 31.** The DoD-96 bit allocation, header, and maximum decimal values

1995

1996

1997

## 4 URI Representation

1998

This section defines standards for the encoding of the Electronic Product Code™ as a

1999

Uniform Resource Identifier (URI). The URI Encoding complements the EPC Tag

2000

Encodings defined for use within RFID tags and other low-level architectural components.

2001

URIs provide a means for application software to manipulate Electronic Product Codes in a

2002

way that is independent of any particular tag-level representation, decoupling application

2003

logic from the way in which a particular Electronic Product Code was obtained from a tag.

2004

*Explanation (non-normative): The pure identity URI for a given EPC is the same regardless of the encoding. For example, the following pure identity URI*

2005

*urn:epc:id:sgtin:0064141.112345.400 is the same regardless of whether it is encoded into a*

2006

*tag as an SGTIN-96 or an SGTIN-198. Other representations than the pure identity URI for*

2007

*use above the reader or middleware layer shall not be used, because they can lead to*

2008

*misinterpretations in the information system. Exclusively on the reader layer and below the*

2009

*encoding schemes including header, filter value and partition must be considered for*

2010

*filtering or writing processes.*

2011

2012

This section defines four categories of URI. The first are URIs for pure identities,

2013

sometimes called “canonical forms.” These contain only the unique information that

2014

identifies a specific physical object, and are independent of tag encodings. The second

2015

category is URIs that represent specific tag encodings. These are used in software

2016

applications where the encoding scheme is relevant, as when commanding software to write

2017

a tag. The third category is URIs that represent patterns, or sets of EPCs. These are used

2018

when instructing software how to filter tag data. The last category is a URI representation

2019

for raw tag information, generally used only for error reporting purposes.

2020

All categories of URIs are represented as Uniform Resource Names (URNs) as defined by

2021

[RFC2141], where the URN Namespace is epc.

2022

This section complements Section 3, EPC Bit-level Encodings, which specifies the currently

2023

defined tag-level representations of the Electronic Product Code.

## 2024 **4.1 URI Forms for Pure Identities**

2025 (This section is non-normative; the formal specifications for the URI types are given in  
2026 Sections 4.2.4 and 5.)

2027 URI forms are provided for pure identities, which contain just the EPC fields that serve to  
2028 distinguish one object from another. These URIs take the form of Uniform Resource Names  
2029 (URNs), with a different URN namespace allocated for each pure identity type.

2030 For the EPC General Identifier (Section 2.1.1), the pure identity URI representation is as  
2031 follows:

2032 `urn:epc:id:gid:GeneralManagerNumber.ObjectClass.SerialNumber`

2033 In this representation, the three fields *GeneralManagerNumber*, *ObjectClass*, and  
2034 *SerialNumber* correspond to the three components of an EPC General Identifier as  
2035 described in Section 2.1.1. In the URI representation, each field is expressed as a decimal  
2036 integer, with no leading zeros (except where a field's value is equal to zero, in which case a  
2037 single zero digit is used).

2038 There are also pure identity URI forms defined for identity types corresponding to certain  
2039 types within the GS1 System family of codes as defined in Section 2.1.2; namely, the  
2040 Serialized Global Trade Item Number (SGTIN), the Serial Shipping Container Code (SSCC),  
2041 the Serialized Global Location Number (SGLN), the Global Reusable Asset Identifier  
2042 (GRAI), the Global Individual Asset Identifier (GIAI), the Global Service Relation Number  
2043 (GSRN) and the Global Document Type Identifier (GDTI). The URI representations  
2044 corresponding to these identifiers are as follows:

2045 `urn:epc:id:sgtin:CompanyPrefix.ItemReference.SerialNumber`

2046 `urn:epc:id:sscc:CompanyPrefix.SerialReference`

2047 `urn:epc:id:sgln:CompanyPrefix.LocationReference.ExtensionComponent`

2048 `urn:epc:id:grai:CompanyPrefix.AssetType.SerialNumber`

2049 `urn:epc:id:giai:CompanyPrefix.IndividualAssetReference`

2050 `urn:epc:id:gsrc:CompanyPrefix.ServiceReference`

2051 `urn:epc:id:gdti:CompanyPrefix.DocumentType.SerialNumber`

2052 In these representations, *CompanyPrefix* corresponds to a GS1 company prefix assigned  
2053 to a manufacturer by GS1. (A UCC company prefix is converted to a GS1 company prefix  
2054 by adding one leading zero at the beginning.) The number of digits in this field is significant,  
2055 and leading zeros are included as necessary.

2056 The *ItemReference*, *SerialReference*, *LocationReference*, *AssetType*,  
2057 *ServiceReference* and *DocumentType* fields correspond to the similar fields of the  
2058 GTIN, SSCC, GLN, GRAI, GSRN and GDTI respectively. Like the *CompanyPrefix*  
2059 field, the number of digits in these fields is significant, and leading zeros are included as  
2060 necessary. The number of digits in these fields, when added to the number of digits in the  
2061 *CompanyPrefix* field, always total the same number of digits according to the identity  
2062 type: 13 digits total for SGTIN, 17 digits total for SSCC, 12 digits total for SGLN, 12  
2063 characters total for the GRAI, 17 digits total for GSRN and 12 characters total for the GDTI.

2064 (The *ItemReference* field of the SGTIN includes the GTIN Indicator (PI) digit,  
2065 appended to the beginning of the item reference. The *SerialReference* field includes  
2066 the SSCC Extension Digit (ED), followed by the serial reference. In no case are check digits  
2067 included in URI representations.)

2068 The *SerialNumber* field of the SGTIN and GRAI, the *ExtensionComponent* of the  
2069 SGLN, as well as the *IndividualAssetReference* field of the GIAI, may include  
2070 digits, letters, and certain other characters. In order for an SGTIN, SGLN, GRAI, or GIAI to  
2071 be encoded on a 96-bit tag, however, these fields must consist only of digits with no leading  
2072 zeros. These restrictions are defined in the encoding procedures for these types, as well as in  
2073 Appendix E.

2074 An SGTIN, SSCC, etc in this form is said to be in SGTIN-URI form, SSCC-URI form, etc  
2075 form, respectively. Here are examples:

2076 urn:epc:id:sgtin:0652642.800031.400

2077 urn:epc:id:sscc:0652642.0123456789

2078 urn:epc:id:sgln:0652642.12345.40 (Use this form when Extension  
2079 Component is used)

2080 urn:epc:id:sgln:0652642.12345.0 (Use this form when Extension  
2081 Component is not used)

2082 urn:epc:id:grai:0652642.12345.1234

2083 urn:epc:id:giai:0652642.123456

2084 urn:epc:id:gsrn:0652642.0123456789

2085 urn:epc:id:gdti:0652642.12345.1234

2086 Referring to the first example, the corresponding GTIN-14 code is 80652642000311. This  
2087 divides as follows: the first digit (8) is the PI digit, which appears as the first digit of the  
2088 *ItemReference* field in the URI, the next seven digits (0652642) are the  
2089 *CompanyPrefix*, the next five digits (00031) are the remainder of the *ItemReference*,  
2090 and the last digit (1) is the check digit, which is not included in the URI.

2091 Referring to the second example, the corresponding SSCC is 006526421234567896 and the  
2092 last digit (6) is the check digit, not included in the URI.

2093 Referring to the third and fourth examples, the corresponding GLN is 0652642123458,  
2094 where the last digit (8) is the check digit, not included in the URI.

2095 Referring to the fifth example, the corresponding GRAI is 06526421234581234. The digit  
2096 (8) which is the check digit and the zero padding digit that is used in the GS1-128 bar code  
2097 representation of the GRAI are not included in the URI.

2098 Referring to the sixth example, the corresponding GIAI is 0652642123456. (GIAI codes do  
2099 not include a check digit.)

2100 Referring to the seventh example, the corresponding GSRN is 065264201234567894, where  
2101 the last digit (4) is the check digit, not included in the URI.

2102 Referring to the eighth example, the corresponding GDTI is 06526421234581234, where the  
2103 digit (8) is the check digit, not included in the URI.

2104 Note that all eight URI forms have an explicit indication of the division between the  
2105 company prefix and the remainder of the code. This is necessary so that the URI  
2106 representation may be converted into tag encodings. In general, the URI representation may  
2107 be converted to the corresponding GS1 numeric form (by combining digits and calculating  
2108 the check digit), but converting from the GS1 numeric form to the corresponding URI  
2109 representation requires independent knowledge of the length of the company prefix.

2110 For the DoD identifier as defined in Section 3.9, the pure identity URI representation is as  
2111 follows:

2112 `urn:epc:id:usdod:CAGECodeOrDODAAC.serialNumber`

2113 where *CAGECodeOrDODAAC* is the five-character CAGE code or six-character DoDAAC,  
2114 and *serialNumber* is the serial number represented as a decimal integer with no leading  
2115 zeros (except that a serial number whose value is zero should be represented as a single zero  
2116 digit). Note that a space character is never included as part of *CAGECodeOrDODAAC* in the  
2117 URI form, even though on a 96-bit tag a space character is used to pad the five-character  
2118 CAGE code to fit into the six-character field on the tag.

2119

## 2120 **4.2 URI Forms for Related Data Types**

2121 (This section is non-normative; the formal specifications for the URI types are given in  
2122 Sections 4.3 and Section 5.)

2123 There are several data types that commonly occur in applications that manipulate Electronic  
2124 Product Codes, which are not themselves Electronic Product Codes but are closely related.  
2125 This specification provides URI forms for those as well. The general form of the *epc* URN  
2126 Namespace is

2127 `urn:epc:type:typeSpecificPart`

2128 The *type* field identifies a particular data type, and *typeSpecificPart* encodes  
2129 information appropriate for that data type. Currently, there are three possibilities defined for  
2130 *type*, discussed in the next three sections.

### 2131 **4.2.1 URIs for EPC Tags**

2132 In some cases, it is desirable to encode in URI form a specific tag encoding of an EPC. For  
2133 example, an application may wish to report to an operator what kinds of tags have been read.  
2134 In another example, an application responsible for programming tags needs to be told not  
2135 only what Electronic Product Code to put on a tag, but also the encoding scheme to be used.  
2136 Finally, applications that wish to manipulate any additional data fields on tags need some  
2137 representation other than the pure identity forms.

2138 EPC Tag URIs are encoded by setting the *type* field to *tag*, with the entire URI having  
2139 this form:

2140 `urn:epc:tag:EncName:EncodingSpecificFields`

2141 where *EncName* is the name of an EPC Tag Encoding scheme, and  
2142 *EncodingSpecificFields* denotes the data fields required by that encoding scheme,  
2143 separated by dot characters. Exactly what fields are present depends on the specific  
2144 encoding scheme used.

2145 In general, there are one or more encoding schemes (and corresponding *EncName* values)  
2146 defined for each pure identity type. For example, the SGTIN Identifier has two encodings  
2147 defined: *sgtin-96* and *sgtin-198*, corresponding to the 96-bit encoding and the 198-  
2148 bit encoding. Note that these encoding scheme names are in one-to-one correspondence with  
2149 unique tag Header values, which are used to represent the encoding schemes on the tag itself.

2150 The *EncodingSpecificFields*, in general, include all the fields of the corresponding  
2151 pure identity type, possibly with additional restrictions on numeric range, plus additional  
2152 fields supported by the encoding. For example, all of the defined encodings for the  
2153 Serialized GTIN include an additional Filter Value that applications use to do tag filtering  
2154 based on object characteristics associated with (but not encoded within) an object's pure  
2155 identity.

2156 Here is an example: a Serialized GTIN 96-bit encoding:

2157 `urn:epc:tag:sgtin-96:3.0652642.800031.400`

2158 In this example, the number 3 is the Filter Value.

2159 The tag URI for the DoD identifier is as follows:

2160 `urn:epc:tag:tagType:filter.CAGECodeOrDODAAC.serialNumber`

2161 where *tagType* is *usdod-96*, *filter* is the filter value represented as one or two  
2162 decimal digits (0-15), and the other two fields are as defined above in 4.1.

2163

## 2164 **4.2.2 URIs for Raw Bit Strings Arising From Invalid Tags**

2165 Certain bit strings do not correspond to legal encodings. Here are several examples:

- 2166 • If the most significant bits of a bit string cannot be recognized as a valid EPC header, the  
2167 bit-level pattern is not a legal EPC Tag Encoding.
- 2168 • If the most significant bits of a bit string are recognized as a valid EPC header, but the  
2169 binary value of a field in the corresponding tag encoding is greater than the value that  
2170 can be contained in the number of decimal digits in that field in the URI form, the bit  
2171 level pattern is not a legal EPC Tag Encoding.
- 2172 • A Gen 2 Tag whose “toggle bit” is set to one (see Section 3.2) by definition does not  
2173 contain an EPC Tag Encoding.

2174 While in these situations a bit string is not a legal EPC Tag Encoding, software may wish to  
2175 report such invalid bit-level patterns to users or to other software. For such cases, a  
2176 representation of invalid bit-level patterns as URIs is provided. The *raw* form of the URI has  
2177 this general form:

2178 urn:epc:raw:BitLength.Value

2179 where *BitLength* is the number of bits in the invalid representation, and *Value* is the  
2180 entire bit-level representation converted to a single hexadecimal number and preceded by the  
2181 letter “x”. For example, this bit string:

2182 0000000000000000000000000100100011010011011110101011011011111011101111

2183 which is invalid because no valid header begins with 0000 0000, corresponds to this raw  
2184 URI:

2185 urn:epc:raw:64.x00001234DEADBEEF

2186 In order to ensure that a given bit string has only one possible raw URI representation, the  
2187 number of digits in the hexadecimal value is required to be equal to the *BitLength* divided  
2188 by four and rounded up to the nearest whole number. Moreover, only uppercase letters are  
2189 permitted for the hexadecimal digits A, B, C, D, E, and F.

2190 It is intended that this URI form be used only when reporting errors associated with reading  
2191 invalid tags and when representing partially written tag. It is *not* intended to be a general  
2192 mechanism for communicating arbitrary bit strings for other purposes.

2193 *Explanation (non-normative): The reason for recommending against using the raw URI for*  
2194 *general purposes is to avoid having an alternative representation for legal tag encodings.*

2195 Earlier versions of this specification described a decimal, as opposed to hexadecimal, version  
2196 of the raw URI. This is still supported for back-compatibility, but its use is no longer  
2197 recommended. The “x” character is included so that software may distinguish between the  
2198 decimal and hexadecimal forms.

#### 2199 **4.2.2.1 Use of the Raw URI with Gen 2 Tags**

2200 The EPC memory of a Gen 2 Tag may contain either an EPC Tag Encoding or a value from  
2201 a different numbering system for which an ISO Application Family Identifier (AFI) has been  
2202 assigned. The “toggle” bit (bit 17x) of EPC memory distinguishes between these two  
2203 possibilities (see Section 3.2).

2204 The Raw URI as described above is intended primarily to represent undecodable EPC Tag  
2205 Encodings or partially written tags. For a Gen 2 Tag, therefore, the Raw URI described  
2206 above is used only when the toggle bit is a zero, indicating that the tag is supposed to contain  
2207 an EPC Tag Encoding.

2208 For completeness, an alternative form of the Raw URI is provided to represent the contents  
2209 of a UHF Class 1 Gen 2 Tag whose toggle bit is a one. It has the following form:

2210 urn:epc:raw:BitLength.AFI.Value

2211 where *BitLength* is the number of bits in the non-EPC representation (not including the  
2212 AFI), AFI is the Application Family Identifier represented as a two-digit hexadecimal  
2213 number and preceded by the letter “x”, and *Value* is the remainder of EPC memory  
2214 converted to a single hexadecimal number and preceded by the letter “x”.

2215 **4.2.2.2 The Length Field of a Raw URI when using Gen 2 Tags (non-normative)**

2216 (This non-normative section explains a subtle interaction between the Raw URI and the  
2217 length indication on Gen 2 Tags.)

2218 Unlike earlier generations of RFID tags, the Gen 2 Tag is designed so that the length of the  
2219 EPC Tag Encoding stored on the tag is not necessarily the same as the total length of EPC  
2220 memory provided. The Gen 2 Specification provides a five-bit length indication, that  
2221 indicates the length of the EPC memory to the nearest multiple of 16 bits (see Section 3.2.2).

2222 Because of the way the EPC Tag Encoding aligns in the Gen 2 Tag's EPC memory, the five-  
2223 bit length indication does not necessarily indicate the length of the EPC Tag Encoding. This  
2224 is because the length indication is limited to expressing multiples of 16 bits, including the  
2225 first 16 bits in the protocol control (PC) bits which is not part of the EPC Tag Encoding. For  
2226 example, if a Gen 2 Tag contains an SGTIN-198 EPC, the EPC Tag Encoding is 198 bits,  
2227 which means there are total of 214 bits is considered when calculating the length indicator  
2228 (198 EPC Tag Encoding bits plus the 16 PC bits). The nearest round up length indicator  
2229 value is 01101 (binary), which indicates a total length of 224 bits. Working in the other  
2230 direction, if a length indicator of 01101 is read from a Gen 2 Tag, it indicates a total of 224  
2231 bits including the 16 PC bits, and therefore appears to indicate an EPC Tag Encoding of 208  
2232 bits.

2233 This does not present a problem when a Gen 2 Tag contains a valid EPC. The procedures in  
2234 Sections 5.3 and 5.4 use the header table in Section 3.1 to determine the length of the EPC,  
2235 and discard any extra bits that may be implied by the length indication. When the contents  
2236 of a Gen 2 Tag are converted to a Raw URI, however, the length indication on the tag is used  
2237 to calculate the length in the URI. Therefore the length representation in the raw URI will  
2238 have different bit length to the EPC Tag Encoding bits. Also one must consider the fact that  
2239 value field in the raw URI may be different, because the values from Gen 2 tags may also  
2240 include excess bits that are filled with zeros up to the word boundary.

2241 For these and other reasons, Raw URIs should never be used within information systems to  
2242 represent valid EPCs.

2243 **4.2.3 URIs for EPC Patterns**

2244 Certain software applications need to specify rules for filtering lists of tags according to  
2245 various criteria. This specification provides a *pattern* URI form for this purpose. A pattern  
2246 URI does not represent a single tag encoding, but rather refers to a set of tag encodings. A  
2247 typical pattern looks like this:

2248 `urn:epc:pat:sgtin-96:3.0652642.[102400-204700].*`

2249 This pattern refers to any EPC SGTIN Identifier 96-bit tag, whose Filter field is 3, whose  
2250 Company Prefix is 0652642, whose Item Reference is in the range  $102400 \leq \textit{itemReference}$   
2251  $\leq 204700$ , and whose Serial Number may be anything at all.

2252 In general, there is a pattern form corresponding to each tag encoding form (Section 4.2.1),  
2253 whose syntax is essentially identical except that ranges or the star (\*) character may be used  
2254 in each field.

2255 For the SGTIN, SSCC, SGLN, GRAI, GIAI, GSRN and GDTI patterns, the pattern syntax  
2256 slightly restricts how wildcards and ranges may be combined. Only two possibilities are  
2257 permitted for the *CompanyPrefix* field. One, it may be a star (\*), in which case the  
2258 following field (*ItemReference*, *SerialReference*, *LocationReference*,  
2259 *AssetType*, *IndividualAssetReference*, *ServiceReference* or  
2260 *DocumentType*) must also be a star. Two, it may be a specific company prefix, in which  
2261 case the following field may be a number, a range, or a star. A range may not be specified  
2262 for the *CompanyPrefix*.

2263 *Explanation (non-normative): Because the company prefix is variable length, a range may*  
2264 *not be specified, as the range might span different lengths. When a particular company*  
2265 *prefix is specified, however, it is possible to match ranges or all values of the following field,*  
2266 *because its length is fixed for a given company prefix. The other case that is allowed is when*  
2267 *both fields are a star, which works for all tag encodings because the corresponding tag*  
2268 *fields (including the Partition field, where present) are simply ignored.*

2269 The pattern URI for the DoD Construct is as follows:

2270 `urn:epc:pat:tagType:filterPat.CAGECodeOrDODAACPat.serialNumber`  
2271 `Pat`

2272 where *tagType* is as defined above in 4.2.1, *filterPat* is either a filter value, a range of  
2273 the form [ *lo-hi* ], or a \* character; *CAGECodeOrDODAACPat* is either a CAGE  
2274 Code/DODAAC or a \* character; and *serialNumberPat* is either a serial number, a  
2275 range of the form [ *lo-hi* ], or a \* character.

#### 2276 **4.2.4 URIs for EPC Pure Identity Patterns**

2277 Certain software applications need to specify rules for filtering lists of EPC pure identities  
2278 according to various criteria. This specification provides a *pure identity pattern* URI form  
2279 for this purpose. A pure identity pattern URI does not represent a single EPC, but rather  
2280 refers to a set of EPCs. A typical pure identity pattern looks like this:

2281 `urn:epc:idpat:sgtin:0652642.*.*`

2282 This pattern refers to any EPC SGTIN, whose Company Prefix is 0652642, whose Item  
2283 Reference and Serial Number may be anything at all. The tag length and filter bits are not  
2284 considered at all in matching the pattern to EPCs.

2285 In general, there is a pattern form corresponding to each pure identity form (Section 4.1),  
2286 whose syntax is essentially identical except any number of fields starting at the right may be  
2287 a star (\*). This is more restrictive than tag patterns (Section 4.2.3), in that the star characters  
2288 must occupy adjacent rightmost fields and the range syntax is not allowed at all.

2289 The pure identity pattern URI for the DoD Construct is as follows:

2290 `urn:epc:idpat:usdod:CAGECodeOrDODAACPat.serialNumberPat`

2291 with similar restrictions on the use of star (\*).

2292 **4.3 Syntax**

2293 The syntax of the EPC-URI and the URI forms for related data types are defined by the  
2294 following grammar.

2295 **4.3.1 Common Grammar Elements**

2296 NumericComponent ::= ZeroComponent | NonZeroComponent

2297 ZeroComponent ::= "0"

2298 NonZeroComponent ::= NonZeroDigit Digit\*

2299 PaddedNumericComponent ::= Digit+

2300 Digit ::= "0" | NonZeroDigit

2301 NonZeroDigit ::= "1" | "2" | "3" | "4"

2302 | "5" | "6" | "7" | "8" | "9"

2303 UpperAlpha ::= "A" | "B" | "C" | "D" | "E" | "F" | "G"

2304 | "H" | "I" | "J" | "K" | "L" | "M" | "N"

2305 | "O" | "P" | "Q" | "R" | "S" | "T" | "U"

2306 | "V" | "W" | "X" | "Y" | "Z"

2307 LowerAlpha ::= "a" | "b" | "c" | "d" | "e" | "f" | "g"

2308 | "h" | "i" | "j" | "k" | "l" | "m" | "n"

2309 | "o" | "p" | "q" | "r" | "s" | "t" | "u"

2310 | "v" | "w" | "x" | "y" | "z"

2311 OtherChar ::= "!" | "'" | "(" | ")" | "\*" | "+" | "," | "-"

2312 | "." | ":" | ";" | "=" | "\_"

2313 UpperHexChar ::= Digit | "A" | "B" | "C" | "D" | "E" | "F"

2314 HexComponent ::= UpperHexChar+

2315 Escape ::= "%" HexChar HexChar

2316 HexChar ::= UpperHexChar | "a" | "b" | "c" | "d" | "e" | "f"

2317 GS3A3Char ::= Digit | UpperAlpha | LowerAlpha | OtherChar

2318 | Escape

2319 GS3A3Component ::= GS3A3Char+

2320 The syntactic construct GS3A3Component is used to represent fields of GS1 codes that  
2321 permit alphanumeric and other characters as specified in Figure 3A3-1 of the GS1 General  
2322 Specifications (see Appendix F). Owing to restrictions on URN syntax as defined by  
2323 [RFC2141], not all characters permitted in the GS1 General Specifications may be  
2324 represented directly in a URN. Specifically, the characters " (double quote), % (percent), &  
2325 (ampersand), / (forward slash), < (less than), > (greater than), and ? (question mark) are  
2326 permitted in the GS1 General Specifications but may not be included directly in a URN. To  
2327 represent one of these characters in a URN, escape notation must be used in which the  
2328 character is represented by a percent sign, followed by two hexadecimal digits that give the  
2329 ASCII character code for the character.

2330 **4.3.2 EPCGID-URI**

2331 EPCGID-URI ::= "urn:epc:id:gid:" 2\*(NumericComponent ".")  
2332 NumericComponent

2333 **4.3.3 SGTIN-URI**

2334 SGTIN-URI ::= "urn:epc:id:sgtin:" SGTINURIBody  
2335 SGTINURIBody ::= 2\*(PaddedNumericComponent ".") GS3A3Component

2336 The number of characters in the two PaddedNumericComponent fields must total 13  
2337 (not including any of the dot characters).

2338 The Serial Number field of the SGTIN-URI is expressed as a GS3A3Component, which  
2339 permits the representation of all characters permitted in the GS1-128 Application Identifier  
2340 21 Serial Number according to the GS1 General Specifications. SGTIN-URIs that are  
2341 derived from 96-bit tag encodings, however, will have Serial Numbers that consist only of  
2342 digits and which have no leading zeros. These limitations are described in the encoding  
2343 procedures, and in Appendix E.

2344 **4.3.4 SSCC-URI**

2345 SSCC-URI ::= "urn:epc:id:sscc:" SSCCURIBody  
2346 SSCCURIBody ::= PaddedNumericComponent "."  
2347 PaddedNumericComponent

2348 The number of characters in the two PaddedNumericComponent fields must total 17  
2349 (not including any of the dot characters).

2350 **4.3.5 SGLN-URI**

2351 SGLN-URI ::= "urn:epc:id:sgln:" SGLNURIBody  
2352 SGLNURIBody ::= 2\*(PaddedNumericComponent ".") GS3A3Component

2353 The number of characters in the two PaddedNumericComponent fields must total 12  
2354 (not including any of the dot characters).

2355 The GLN *Extension Component* field of the SGLN-URI is expressed as a  
2356 GS3A3Component, which permits the representation of all characters permitted in the  
2357 GS1-128 Application Identifier 254 Extension Component according to the GS1 General  
2358 Specifications. SGLN-URIs that are derived from 96-bit tag encodings, however, will have  
2359 Extension Component that consist only of digits and which have no leading zeros. These  
2360 limitations are described in the encoding procedures, and in Appendix E

2361 **4.3.6 GRAI-URI**

2362 GRAI-URI ::= "urn:epc:id:grai:" GRAIURIBody  
2363 GRAIURIBody ::= 2\*(PaddedNumericComponent ".") GS3A3Component

2364 The number of characters in the two `PaddedNumericComponent` fields must total 12  
2365 (not including any of the dot characters).

2366 The Serial Number field of the GRAI-URI is expressed as a `GS3A3Component`, which  
2367 permits the representation of all characters permitted in the Serial Number field of the GRAI  
2368 according to the GS1 General Specifications. GRAI-URIs that are derived from 96-bit tag  
2369 encodings, however, will have Serial Numbers that consist only of digit characters and which  
2370 have no leading zeros. These limitations are described in the encoding procedures, and in  
2371 Appendix E.

#### 2372 **4.3.7 GIAI-URI**

2373 `GIAI-URI ::= "urn:epc:id:giai:" GIAIURIBody`

2374 `GIAIURIBody ::= PaddedNumericComponent "." GS3A3Component`

2375 The total number of characters in the `PaddedNumericComponent` and  
2376 `GS3A3Component` fields must not exceed 30 (not including the dot character that separates  
2377 the two fields).

2378 The Individual Asset Reference field of the GIAI-URI is expressed as a `GS3A3Component`,  
2379 which permits the representation of all characters permitted in the Individual Asset  
2380 Reference field of the GIAI according to the GS1 General Specifications. GIAI-URIs that is  
2381 derived from 96-bit tag encodings, however, will have Individual Asset References that  
2382 consist only of digit characters and which have no leading zeros. These limitations are  
2383 described in the encoding procedures, and in Appendix E.

2384

#### 2385 **4.3.8 GSRN-URI**

2386 `GSRN-URI ::= "urn:epc:id:gsrc:" GSRNURIBody`

2387 `GSRNURIBody ::= PaddedNumericComponent "."`

2388 `PaddedNumericComponent`

2389 The number of characters in the two `PaddedNumericComponent` fields must total 17  
2390 (not including any of the dot characters).

2391

2392

#### 2393 **4.3.9 GDTI-URI**

2394 `GDTI-URI ::= "urn:epc:id:gdti:" GDTIURIBody`

2395 `GDTIURIBody ::= 2*(PaddedNumericComponent ".")`

2396 `PaddedNumericComponent`

2397 The number of characters in the first two `PaddedNumericComponent` fields must total  
2398 12 (not including any of the dot characters).

2399 The third field, which is the Serial Number field of the GDTI-URI is expressed as a  
2400 PaddedNumericComponent, which permits the representation of all numeric characters.  
2401 GDTI-URIs that are derived from 113-bit tag encodings allow Serial Numbers that consist  
2402 only of digits but allow leading zeros. GDTI-URIs that are derived from 96-bit tag encodings,  
2403 however, will have Serial Numbers that consist only of digits and which have no leading  
2404 zeros. These limitations are described in the encoding procedures, and in Appendix E.

#### 2405 **4.3.10 EPC Tag URI**

2406 TagURI ::= "urn:epc:tag:" TagURIBody

2407 TagURIBody ::= GIDTagURIBody | SGTINSGLNGRAI96TagURIBody |  
2408 SGTINSGLNGRAIAlphaTagURIBody | SSCCTagURIBody |  
2409 GIAI96TagURIBody | GIAIAlphaTagURIBody | GSRNTagURIBody |  
2410 GDTITagURIBody

2411 GIDTagURIBody ::= GIDTagEncName ":" 2\*(NumericComponent ".")  
2412 NumericComponent

2413 GIDTagEncName ::= "gid-96"

2414 SGTINSGLNGRAITag96URIBody ::= SGTINSGLNGRAI96TagEncName ":"  
2415 NumericComponent "." 2\*(PaddedNumericComponent ".")  
2416 NumericComponent

2417 SGTINSGLNGRAITagAlphaURIBody ::= SGTINSGLNGRAIAlphaTagEncName  
2418 ":" NumericComponent "." 2\*(PaddedNumericComponent ".")  
2419 GS3A3Component

2420 SGTINSGLNGRAI96TagEncName ::= "sgtin-96" | "sgln-96" | "grai-  
2421 96"

2422 SGTINSGLNGRAIAlphaTagEncName ::= "sgtin-198" | "sgln-195" |  
2423 "grai-170"

2424 SSCCTagURIBody ::= SSCCTagEncName ":" NumericComponent 2\*(".")  
2425 PaddedNumericComponent)

2426 SSCCTagEncName ::= "sscc-96"

2427 GIAI96TagURIBody ::= GIAI96TagEncName ":" NumericComponent ".")  
2428 PaddedNumericComponent ".") NumericComponent

2429 GIAIAlphaTagURIBody ::= GIAIAlphaTagEncName ":"  
2430 NumericComponent ".") PaddedNumericComponent ".") GS3A3Component

2431 GIAI96TagEncName ::= "giai-96"

2432 GIAIAlphaTagEncName ::= "giai-202"

2433 GSRNTagURIBody ::= GSRNTagEncName ":" NumericComponent 2\*(".")  
2434 PaddedNumericComponent)

2435 GSRNTagEncName ::= "gsrn-96"

2436 GDTITagURIBody ::= GDTITagEncName ":"NumericComponent 3\*("."  
2437 PaddedNumericComponent)  
2438 GDTITagEncName ::= "gdti-96" | "gdti-113"  
2439

#### 2440 **4.3.11 Raw Tag URI**

2441 RawURI ::= "urn:epc:raw:" RawURIBody  
2442 RawURIBody ::= DecimalRawURIBody | HexRawURIBody |  
2443 AFIRawURIBody)  
2444 DecimalRawURIBody ::= NonZeroComponent "." NumericComponent  
2445 HexRawURIBody ::= NonZeroComponent ".x" HexComponent  
2446 AFIRawURIBody ::= NonZeroComponent ".x" HexComponent ".x"  
2447 HexComponent

#### 2448 **4.3.12 EPC Pattern URI**

2449 PatURI ::= "urn:epc:pat:" PatBody  
2450 PatBody ::= GIDPatURIBody | SGTINSGLNGRAI96PatURIBody |  
2451 SGTINSGLNGRAIAlphaPatURIBody | SSCCPatURIBody |  
2452 GIAI96PatURIBody | GIAIAlphaPatURIBody | GSRNPatURIBody |  
2453 GDTIPatURIBody  
2454 GIDPatURIBody ::= GIDTagEncName ":" 2\*(PatComponent ".")  
2455 PatComponent  
2456 SGTINSGLNGRAI96PatURIBody ::= SGTINSGLNGRAI96TagEncName ":"  
2457 PatComponent "." GS1PatBody "." PatComponent  
2458 SGTINSGLNGRAIAlphaPatURIBody ::= SGTINSGLNGRAIAlphaTagEncName  
2459 ":" PatComponent "." GS1PatBody "." GS3A3PatComponent  
2460 SSCCPatURIBody ::= SSCCTagEncName ":" PatComponent "."  
2461 GS1PatBody  
2462 GIAI96PatURIBody ::= GIAI96TagEncName ":" PatComponent "."  
2463 GS1PatBody  
2464 GIAIAlphaPatURIBody ::= GIAIAlphaTagEncName ":" PatComponent  
2465 "." GS1GS3A3PatBody  
2466 GSRNPatURIBody ::= GSRNTagEncName ":" PatComponent "."  
2467 GS1PatBody  
2468 GDTIPatURIBody ::= GDTI96PatURIBody | GDTI113PatURIBody  
2469 GDTI96PatURIBody ::= "GDTI-96:" PatComponent "."GS1PatBody "."  
2470 PatComponent

```

2471 GDTI113PatURIBody ::= "GDTI-113:" PatComponent "."GS1PatBody
2472 "." NumericOrStarComponent
2473 NumericOrStarComponent ::= NumericComponent | StarComponent
2474 GS1PatBody ::= "*.*" | ( PaddedNumericComponent "."
2475 PatComponent )
2476 GS1GS3A3PatBody ::= "*.*" | ( PaddedNumericComponent "."
2477 GS3A3PatComponent )
2478 PatComponent ::= NumericComponent
2479 | StarComponent
2480 | RangeComponent
2481 GS3A3PatComponent ::= GS3A3Component | StarComponent
2482 StarComponent ::= "*"
2483 RangeComponent ::= "[" NumericComponent "-"
2484 NumericComponent "]"
2485 For a RangeComponent to be legal, the numeric value of the first NumericComponent
2486 must be less than or equal to the numeric value of the second NumericComponent.

```

### 2487 **4.3.13 EPC Identity Pattern URI**

```

2488 IDPatURI ::= "urn:epc:idpat:" IDPatBody
2489 IDPatBody ::= GIDIDPatURIBody | SGTINIDPatURIBody |
2490 SGLNIDPatURIBody | GIAIIDPatURIBody | SSCCIDPatURIBody |
2491 GRAIIDPatURIBody | GSRNIDPatURIBody | GDTIIDPatURIBody
2492 GIDIDPatURIBody ::= "gid:" GIDIDPatURIMain
2493 GIDIDPatURIMain ::=
2494 2*(NumericComponent ".") NumericComponent
2495 | 2*(NumericComponent ".") "*"
2496 | NumericComponent ".*.*"
2497 | ".*.*"
2498 SGTINIDPatURIBody ::= "sgtin:" SGTINSGLNGRAIIDPatURIMain
2499 GRAIIDPatURIBody ::= "grai:" SGTINSGLNGRAIIDPatURIMain
2500 SGLNIDPatURIBody ::= "sgln:" SGTINSGLNGRAIIDPatURIMain
2501 SGTINSGLNGRAIIDPatURIMain ::=
2502 2*(PaddedNumericComponent ".") GS3A3Component
2503 | 2*(PaddedNumericComponent ".") "*"
2504 | PaddedNumericComponent ".*.*"
2505 | ".*.*"
2506 SSCCIDPatURIBody ::= "sscc:" SSCCIDPatURIMain

```

```

2507 SSCCIDPatURIMain ::=
2508     PaddedNumericComponent "." PaddedNumericComponent
2509     | PaddedNumericComponent "*"
2510     | "*.*"
2511 GIAIIDPatURIBody ::= "giai:" GIAIIDPatURIMain
2512 GIAIIDPatURIMain ::=
2513     PaddedNumericComponent "." GS3A3Component
2514     | PaddedNumericComponent "*"
2515     | "*.*"
2516 GSRNIDPatURIBody ::= "gsrn:" GSRNIDPatURIMain
2517 GSRNIDPatURIMain ::=
2518     PaddedNumericComponent "." PaddedNumericComponent
2519     | PaddedNumericComponent "*"
2520     | "*.*"
2521 GDTIIDPatURIBody ::= "gdti:" GDTIIDPatURIMain
2522 GDTIIDPatURIMain ::=
2523     2*(PaddedNumericComponent ".") PaddedNumericComponent
2524     | 2*(PaddedNumericComponent ".") "*"
2525     | PaddedNumericComponent ".*.*"
2526     | ".*.*"
2527
2528
2529 4.3.14 DoD Construct URI
2530 DOD-URI ::= "urn:epc:id:usdod:" CAGECodeOrDODAAC "."
2531 DoDSerialNumber
2532 DODTagURI ::= "urn:epc:tag:" DoDTagType ":" DoDFilter "."
2533 CAGECodeOrDODAAC "." DoDSerialNumber
2534 DODPatURI ::= "urn:epc:pat:" DoDTagType ":" DoDFilterPat "."
2535 CAGECodeOrDODAACPat "." DoDSerialNumberPat
2536 DODIDPatURI ::= "urn:epc:idpat:usdod:" DODIDPatMain
2537 DODIDPatMain ::=
2538     CAGECodeOrDODAAC "." DoDSerialNumber
2539     | CAGECodeOrDODAAC "*"
2540     | "*.*"
2541 DoDTagType ::= "usdod-96"
2542 DoDFilter ::= NumericComponent
2543 CAGECodeOrDODAAC ::= CAGECode | DODAAC
2544 CAGECode ::= CAGECodeOrDODAACChar*5

```

2545 DODAAC ::= CAGECodeOrDODAACChar\*6  
 2546 DoDSerialNumber ::= NumericComponent  
 2547 DoDFilterPat ::= PatComponent  
 2548 CAGECodeOrDODAACPat ::= CAGECodeOrDODAAC | StarComponent  
 2549 DoDSerialNumberPat ::= PatComponent  
 2550 CAGECodeOrDODAACChar ::= Digit | "A" | "B" | "C" | "D" | "E" |  
 2551 "F" | "G" | "H" | "J" | "K" | "L" | "M" | "N" | "P" | "Q" |  
 2552 "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"  
 2553

### 2554 **4.3.15 Summary (non-normative)**

2555 The syntax rules above can be summarized informally as follows:

2556 urn:epc:id:gid:MMM.CCC.SSS  
 2557 urn:epc:id:sgtin:PPP.III.AAA  
 2558 urn:epc:id:sscc:PPP.III  
 2559 urn:epc:id:sgln:PPP.III.AAA  
 2560 urn:epc:id:grai:PPP.III.AAA  
 2561 urn:epc:id:giai:PPP.AAA  
 2562 urn:epc:id:gsrn:PPP.III  
 2563 urn:epc:id:gdti:PPP.III.DDD  
 2564 urn:epc:id:usdod:TTT.SSS  
 2565  
 2566 urn:epc:tag:gid-96:MMM.CCC.SSS  
 2567 urn:epc:tag:sgtin-96:FFF.PPP.III.SSS  
 2568 urn:epc:tag:sgtin-198:FFF.PPP.III.AAA  
 2569 urn:epc:tag:sscc-96:FFF.PPP.III  
 2570 urn:epc:tag:sgln-96:FFF.PPP.III.SSS  
 2571 urn:epc:tag:sgln-195:FFF.PPP.III.AAA  
 2572 urn:epc:tag:grai-96:FFF.PPP.III.SSS  
 2573 urn:epc:tag:grai-170:FFF.PPP.III.AAA  
 2574 urn:epc:tag:giai-96:FFF.PPP.SSS  
 2575 urn:epc:tag:giai-202:FFF.PPP.AAA  
 2576 urn:epc:tag:gsrn-96:FFF.PPP.III  
 2577 urn:epc:tag:gdti-96:FFF.PPP.III.SSS

2578 urn:epc:tag:gdti-113:FFF.PPP.III.DDD  
2579 urn:epc:tag:usdod-96:FFF.TTT.SSS  
2580  
2581 urn:epc:raw:LLL.BBB  
2582 urn:epc:raw:LLL.HHH  
2583 urn:epc:raw:LLL.HHH.HHH  
2584  
2585 urn:epc:idpat:gid:MMM.CCC.SSS  
2586 urn:epc:idpat:gid:MMM.CCC.\*  
2587 urn:epc:idpat:gid:MMM.\*.\*  
2588 urn:epc:idpat:gid:\*. \*.\*  
2589 urn:epc:idpat:sgtin:PPP.III.AAA  
2590 urn:epc:idpat:sgtin:PPP.III.\*  
2591 urn:epc:idpat:sgtin:PPP.\*.\*  
2592 urn:epc:idpat:sgtin:\*. \*.\*  
2593 urn:epc:idpat:sscc:PPP.III  
2594 urn:epc:idpat:sscc:PPP.\*  
2595 urn:epc:idpat:sscc:\*. \*  
2596 urn:epc:idpat:sgln:PPP.III.AAA  
2597 urn:epc:idpat:sgln:PPP.III.\*  
2598 urn:epc:idpat:sgln:PPP.\*.\*  
2599 urn:epc:idpat:sgln:\*. \*.\*  
2600 urn:epc:idpat:grai:PPP.III.AAA  
2601 urn:epc:idpat:grai:PPP.III.\*  
2602 urn:epc:idpat:grai:PPP.\*.\*  
2603 urn:epc:idpat:grai:\*. \*.\*  
2604 urn:epc:idpat:giai:PPP.AAA  
2605 urn:epc:idpat:giai:PPP.\*  
2606 urn:epc:idpat:giai:\*. \*  
2607 urn:epc:idpat:gsrn:PPP.III  
2608 urn:epc:idpat:gsrn:PPP.\*  
2609 urn:epc:idpat:gsrn:\*. \*  
2610 urn:epc:idpat:gdti:PPP.III.DDD

2611 urn:epc:idpat:gdti:PPP.III.\*  
2612 urn:epc:idpat:gdti:PPP.\*.\*  
2613 urn:epc:idpat:gdti:\*.\*. \*  
2614 urn:epc:idpat:usdod:TTT.SSS  
2615 urn:epc:idpat:usdod:TTT.\*  
2616 urn:epc:idpat:usdod:\*. \*  
2617  
2618 urn:epc:pat:gid-96:MMMpat.CCCpat.SSSpat  
2619 urn:epc:pat:sgtin-96:FFFpat.PPP.IIIpat.SSSpat  
2620 urn:epc:pat:sgtin-96:FFFpat.\*.\*.SSSpat  
2621 urn:epc:pat:sgtin-198:FFFpat.PPP.IIIpat.AAApat  
2622 urn:epc:pat:sgtin-198:FFFpat.\*.\*.AAAp  
2623 urn:epc:pat:sscc-96:FFFpat.PPP.IIIpat  
2624 urn:epc:pat:sscc-96:FFFpat.\*.\*  
2625 urn:epc:pat:sgln-96:FFFpat.PPP.IIIpat.SSSpat  
2626 urn:epc:pat:sgln-96:FFFpat.\*.\*.SSSpat  
2627 urn:epc:pat:sgln-195:FFFpat.PPP.IIIpat.AAApat  
2628 urn:epc:pat:sgln-195:FFFpat.\*.\*.AAAp  
2629 urn:epc:pat:grai-96:FFFpat.PPP.IIIpat.SSSpat  
2630 urn:epc:pat:grai-96:FFFpat.\*.\*.SSSpat  
2631 urn:epc:pat:grai-170:FFFpat.PPP.IIIpat.AAApat  
2632 urn:epc:pat:grai-170:FFFpat.\*.\*.AAAp  
2633 urn:epc:pat:giai-96:FFFpat.PPP.SSSpat  
2634 urn:epc:pat:giai-96:FFFpat.\*.\*  
2635 urn:epc:pat:giai-202:FFFpat.PPP.AAApat  
2636 urn:epc:pat:giai-202:FFFpat.\*.\*  
2637 urn:epc:pat:gsrc-96:FFFpat.PPP.IIIpat  
2638 urn:epc:pat:gsrc-96:FFFpat.\*.\*  
2639 urn:epc:pat:gdti-96:FFFpat.PPP.IIIpat.SSSpat  
2640 urn:epc:pat:gdti-96:FFFpat.\*.\*.SSSpat  
2641 urn:epc:pat:gdti-113:FFFpat.PPP.IIIpat.DDDpat  
2642 urn:epc:pat:gdti-113:FFFpat.\*.\*.DDDpat  
2643 urn:epc:pat:usdod-96:FFFpat.TTT.SSSpat

2644 urn:epc:pat:usdod-96:FFFpat.\*.SSSpat  
 2645 where  
 2646 *MMM* denotes a General Manager Number  
 2647 *CCC* denotes an Object Class number  
 2648 *SSS* denotes a numeric Serial Number or GIAI Individual Asset Reference  
 2649 *AAA* denotes an alphanumeric Serial Number or GIAI Individual Asset reference  
 2650 *DDD* denotes a numeric Serial Number that may include leading zeros  
 2651 *PPP* denotes a GS1 Company Prefix  
 2652 *TTT* denotes a US DoD assigned CAGE code or DODAAC  
 2653 *III* denotes an SGTIN Item Reference (prefixed by the Indicator Digit), an SSCC  
 2654 Shipping Container Serial Number (prefixed by the Extension Digit (ED)), a SGLN Location  
 2655 Reference, a GRAI Asset Type, a GSRN Service Relation Number or a GDTI Document  
 2656 Type.  
 2657 *FFF* denotes a filter code as used by the SGTIN, SSCC, SGLN, GRAI, GIAI, GSRN,  
 2658 GDTI and DoD tag encodings  
 2659 *XXXpat* is the same as *XXX* but allowing \* and [lo-hi] pattern syntax in addition  
 2660 (exception: [lo-hi] syntax is not allowed for *AAAp* or *DDDpat*).  
 2661 *LLL* denotes the number of bits of an uninterpreted bit sequence  
 2662 *BBB* denotes the literal value of an uninterpreted bit sequence converted to decimal  
 2663 *HHH* denotes the literal value of an uninterpreted bit sequence converted to hexadecimal  
 2664 and preceded by the character 'x'.  
 2665 and where all numeric fields are in decimal with no leading zeros (unless the overall value of  
 2666 the field is zero, in which case it is represented with a single 0 character), with the exception  
 2667 of the hexadecimal raw representation and *DDD*.  
 2668 Exceptions:  
 2669 1. The length of *PPP* and *III* is significant, and leading zeros are used as necessary.  
 2670 The length of *PPP* is the length of the company prefix as assigned by GS1. The  
 2671 length of *III* plus the length of *PPP* must equal 13 for SGTIN, 17 for SSCC, 12 for  
 2672 GLN, 12 for GRAI, 17 for GSRN or 12 for GDTI.  
 2673 2. The Value field of urn:epc:raw is expressed in hexadecimal if the value is  
 2674 preceded by the character 'x'.

## 2675 **5 Translation between EPC-URI and Other EPC** 2676 **Representations**

2677 This section defines the semantics of EPC-URI encodings, by defining how they are  
 2678 translated into other EPC representations and vice versa.

## 2679 **5.1 Bit string into EPC-URI (pure identity)**

2680 The following procedure translates a bit-level encoding into an EPC-URI:

- 2681 1. Determine the identity type and encoding scheme by finding the row in Table 1  
2682 (Section 3.1) that matches the most significant bits of the bit string. If the most  
2683 significant bits do not match any row of the table, stop: the bit string is invalid and  
2684 cannot be translated into an EPC-URI. If the encoding scheme indicates one of the  
2685 DoD Tag Data Constructs, consult the appropriate U.S. Department of Defense  
2686 document for specific encoding and decoding rules. Otherwise, if the encoding  
2687 scheme is SGTIN-96 or SGTIN-198, proceed to Step 2; if the encoding scheme is  
2688 SSCC-96, proceed to Step 5; if the encoding scheme is SGLN-96 or SGLN-195,  
2689 proceed to Step 8; if the encoding scheme is GRAI-96 or GRAI-170, proceed to  
2690 Step 11; if the encoding scheme is GIAI-96 or GIAI-202, proceed to Step 14; if the  
2691 encoding scheme is GSRN-96, proceed to Step 17; if the encoding scheme is GDTI-  
2692 96 or GDTI-113, proceed to Step 20; if the encoding scheme is GID-96, proceed to  
2693 Step 23.
- 2694 2. Follow the decoding procedure given in Section 3.5.1.2 (for SGTIN-96) or in  
2695 Section 3.5.2.2 (for SGTIN-198) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ , the  
2696 decimal Item Reference and Indicator  $i_1i_2\dots i_{(13-L)}$ , and the Serial Number  $S$ . If the  
2697 decoding procedure fails, stop: the bit-level encoding cannot be translated into an  
2698 EPC-URI.
- 2699 3. Create an EPC-URI by concatenating the following: the string  
2700 `urn:epc:id:sgtin:`, the Company Prefix  $p_1p_2\dots p_L$  where each digit (including  
2701 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2702 character, the Item Reference and Indicator  $i_1i_2\dots i_{(13-L)}$  (handled similarly), a dot (.)  
2703 character, and the Serial Number  $S$  as a decimal integer (SGTIN-96) or alphanumeric  
2704 character (SGTIN-198). For SGTIN-96 the portion corresponding to the Serial  
2705 Number must have no leading zeros, except where the Serial Number is itself zero in  
2706 which case the corresponding URI portion must consist of a single zero character.
- 2707 4. Go to Step 25.
- 2708 5. Follow the decoding procedure given in Section 3.6.1.2 (for SSCC-96) to obtain the  
2709 decimal Company Prefix  $p_1p_2\dots p_L$ , and the decimal Serial Reference  $s_1s_2\dots s_{(17-L)}$ . If  
2710 the decoding procedure fails, stop: the bit-level encoding cannot be translated into an  
2711 EPC-URI.
- 2712 6. Create an EPC-URI by concatenating the following: the string  
2713 `urn:epc:id:sscc:`, the Company Prefix  $p_1p_2\dots p_L$  where each digit (including  
2714 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2715 character, and the Serial Reference  $s_1s_2\dots s_{(17-L)}$  (handled similarly).
- 2716 7. Go to Step 25.
- 2717 8. Follow the decoding procedure given in Section 3.7.1.2 (for SGLN-96) or in Section  
2718 3.7.2.2 (for SGLN-195) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ , the decimal  
2719 Location Reference  $i_1i_2\dots i_{(12-L)}$ , and the Extension Component  $S$ . If the decoding  
2720 procedure fails, stop: the bit-level encoding cannot be translated into an EPC-URI.

- 2721 9. Create an EPC-URI by concatenating the following: the string  
2722 urn:epc:id:sgln:, the Company Prefix  $p_1p_2...p_L$  where each digit (including  
2723 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2724 character, for  $L < 12$  the Location Reference,  $i_1i_2...i_{(12-L)}$  (handled similarly), a dot  
2725 (.) character, and the Extension Component  $S$  as a decimal integer (SGLN-96) or  
2726 alphanumeric character (SGLN-195). For SGLN-96 the portion corresponding to the  
2727 Extension Component must have no leading zeros, except where the Extension  
2728 Component is itself zero in which case the corresponding URI portion must consist of  
2729 a single zero character. If a Location Reference does not exist (where  $L = 12$ ), leave  
2730 no blank space between the two dot (.) characters.
- 2731 10. Go to Step 25.
- 2732 11. Follow the decoding procedure given in Section 3.8.1.2 (for GRAI-96) or in Section  
2733 3.8.2.2 (for GRAI-170) to obtain the decimal Company Prefix  $p_1p_2...p_L$ , the decimal  
2734 Asset Type  $i_1i_2...i_{(12-L)}$ , and the Serial Number  $S$ . If the decoding procedure fails,  
2735 stop: the bit-level encoding cannot be translated into an EPC-URI.
- 2736 12. Create an EPC-URI by concatenating the following: the string  
2737 urn:epc:id:grai:, the Company Prefix  $p_1p_2...p_L$  where each digit (including  
2738 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2739 character, for  $L < 12$  the Asset Type  $i_1i_2...i_{(12-L)}$  (handled similarly), a dot (.)  
2740 character, and the Serial Number  $S$  as a decimal integer (GRAI-96) or alphanumeric  
2741 character (GRAI-170). For GRAI-96 the portion corresponding to the Serial Number  
2742 must have no leading zeros, except where the Serial Number is itself zero in which  
2743 case the corresponding URI portion must consist of a single zero character. If an  
2744 Asset Type does not exist (where  $L = 12$ ), leave no blank space between the two dot  
2745 (.) characters.
- 2746 13. Go to Step 25.
- 2747 14. Follow the decoding procedure given in Section 3.9.1.2 (for GIAI-96) or in 3.9.2.2  
2748 (for GIAI-202) to obtain the decimal Company Prefix  $p_1p_2...p_L$ , and the Individual  
2749 Asset Reference  $S$ . If the decoding procedure fails, stop: the bit-level encoding  
2750 cannot be translated into an EPC-URI.
- 2751 15. Create an EPC-URI by concatenating the following: the string  
2752 urn:epc:id:giai:, the Company Prefix  $p_1p_2...p_L$  where each digit (including  
2753 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2754 character, and the Individual Asset Reference  $S$  as a decimal integer (GIAI-96) or  
2755 alphanumeric character (GIAI-202). For GIAI-96 the portion corresponding to the  
2756 Individual Asset Reference must have no leading zeros, except where the Individual  
2757 Asset Reference is itself zero in which case the corresponding URI portion must  
2758 consist of a single zero character.
- 2759 16. Go to Step 25.
- 2760 17. Follow the decoding procedure given in Section 3.10.1.2 (for GSRN-96) to obtain the  
2761 decimal Company Prefix  $p_1p_2...p_L$ , and the decimal Service Reference  $s_1s_2...s_{(17-L)}$ . If

- 2762 the decoding procedure fails, stop: the bit-level encoding cannot be translated into an  
2763 EPC-URI.
- 2764 18. Create an EPC-URI by concatenating the following: the string  
2765 `urn:epc:id:gsrn:`, the Company Prefix  $p_1p_2...p_L$  where each digit (including  
2766 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2767 character, and the Service Reference  $i_1i_2...i_{(17-L)}$  (handled similarly).
- 2768 19. Go to Step 25
- 2769 20. Follow the decoding procedure given in Section 3.11.1.2 (for GDTI-96) or in  
2770 Section 3.11.2.2 (for GDTI-113) to obtain the decimal Company Prefix  $p_1p_2...p_L$ , the  
2771 decimal Document Type  $i_1i_2...i_{(12-L)}$ , and the Serial Number  $d_{14}d_{15}...d_K$ . If the  
2772 decoding procedure fails, stop: the bit-level encoding cannot be translated into an  
2773 EPC-URI.
- 2774 21. Create an EPC-URI by concatenating the following: the string  
2775 `urn:epc:id:gdti:`, the Company Prefix  $p_1p_2...p_L$  where each digit (including  
2776 any leading zeros) becomes the corresponding ASCII digit character, a dot (.)  
2777 character, the Document Type  $i_1i_2...i_{(12-L)}$  (handled similarly), a dot (.) character, and  
2778 the Serial Number  $d_{14}d_{15}...d_K$ .
- 2779 22. Go to Step 25.
- 2780 23. Follow the decoding procedure given in Section 3.4.1.2 to obtain the General  
2781 Manager Number  $M$ , the Object Class  $C$ , and the Serial Number  $S$ .
- 2782 24. Create an EPC-URI by concatenating the following: the string  
2783 `urn:epc:id:gid:`, the General Manager Number as a decimal integer, a dot (.)  
2784 character, the Object Class as a decimal integer, a dot (.) character, and the Serial  
2785 Number  $S$  as a decimal integer. Each decimal number must have no leading zeros,  
2786 except where the integer is itself zero in which case the corresponding URI portion  
2787 must consist of a single zero character.
- 2788 25. The translation is now complete.

## 2789 5.2 Bit String into Tag or Raw URI

2790 The following procedure translates a bit string of N bits into either an EPC Tag URI or a  
2791 Raw Tag URI:

- 2792 1. Determine the identity type, encoding scheme, and encoding length (K) by finding  
2793 the row in Table 1 (Section 3.1) that matches the most significant bits of the bit string.  
2794 If  $N < K$ , proceed to Step 20; otherwise, continue with the remainder of this  
2795 procedure, using the most significant K bits of the bit string. If the encoding scheme  
2796 indicates one of the DoD Tag Data Constructs, consult the appropriate U.S.  
2797 Department of Defense document for specific encoding and decoding rules. If the  
2798 encoding scheme is SGTIN-96 or SGTIN-198, proceed to Step 2; if the encoding  
2799 scheme is SSCC-96, proceed to Step 5; if the encoding scheme is SGLN-96 or  
2800 SGLN-195, proceed to Step 8; if the encoding scheme is GRAI-96 or GRAI-170,  
2801 proceed to Step 11, if the encoding scheme is GIAI-96 or GIAI-202, proceed to Step

- 2802 14; if the encoding scheme is GSRN-96, proceed to Step 17; if the encoding scheme  
2803 is GDTI-96 or GDTI-113, proceed to Step 20; if the encoding scheme is GID-96,  
2804 proceed to Step 23; otherwise, proceed to Step 26.
- 2805 2. Follow the decoding procedure given in Section 3.5.1.2 (for SGTIN-96) or 3.5.2.2  
2806 (for SGTIN-198) to obtain the decimal Company Prefix  $p_1p_2...p_L$ , the decimal Item  
2807 Reference and Indicator  $i_1i_2...i_{(13-L)}$ , the Filter Value  $F$ , and the Serial Number  $S$ . If  
2808 the decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2809 3. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,  
2810 the encoding scheme (`sgtin-96` or `sgtin-198`), a colon (`:`) character, the Filter  
2811 Value  $F$  as a decimal integer, a dot (`.`) character, the Company Prefix  $p_1p_2...p_L$  where  
2812 each digit (including any leading zeros) becomes the corresponding ASCII digit  
2813 character, a dot (`.`) character, the Item Reference and Indicator  $i_1i_2...i_{(13-L)}$  (handled  
2814 similarly), a dot (`.`) character, and the Serial Number  $S$  as a decimal integer (SGTIN-  
2815 96) or alphanumeric character (SGTIN-198). For SGTIN-96 the portions  
2816 corresponding to the Filter Value and Serial Number must have no leading zeros,  
2817 except where the corresponding integer is itself zero in which case a single zero  
2818 character is used.
- 2819 4. Go to Step 27.
- 2820 5. Follow the decoding procedure given in Section 3.6.1.2 (for SSCC-96) to obtain the  
2821 decimal Company Prefix  $p_1p_2...p_L$ , and the decimal Serial Reference  $i_1i_2...i_{(17-L)}$ , and  
2822 the Filter Value  $F$ . If the decoding procedure fails, proceed to Step 20, otherwise  
2823 proceed to the next step.
- 2824 6. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,  
2825 the encoding scheme (`sscc-96`), a colon (`:`) character, the Filter Value  $F$  as a  
2826 decimal integer, a dot (`.`) character, the Company Prefix  $p_1p_2...p_L$  where each digit  
2827 (including any leading zeros) becomes the corresponding ASCII digit character, a dot  
2828 (`.`) character, and the Serial Reference  $i_1i_2...i_{(17-L)}$  (handled similarly).
- 2829 7. Go to Step 27.
- 2830 8. Follow the decoding procedure given in Section 3.7.1.2 (for SGLN-96) or Section  
2831 3.7.2.2 (for SGLN-195) to obtain the decimal Company Prefix  $p_1p_2...p_L$ , the decimal  
2832 Location Reference  $i_1i_2...i_{(12-L)}$ , the Filter Value  $F$ , and the Extension Component  $S$ .  
2833 If the decoding procedure fails, proceed to Step 20, otherwise proceed to the next  
2834 step.
- 2835 9. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,  
2836 the encoding scheme (`sgln-96` or `sgln-195`), a colon (`:`) character, the Filter  
2837 Value  $F$  as a decimal integer, a dot (`.`) character, the Company Prefix  $p_1p_2...p_L$  where  
2838 each digit (including any leading zeros) becomes the corresponding ASCII digit  
2839 character, a dot (`.`) character, when  $L < 12$  the Location Reference  $i_1i_2...i_{(12-L)}$   
2840 (handled similarly), a dot (`.`) character, and the Extension Component  $S$  as a decimal  
2841 integer (SGLN-96) or alphanumeric character (SGLN-198). For SGLN-96 the  
2842 portions corresponding to the Filter Value and Extension Component must have no  
2843 leading zeros, except where the corresponding integer is itself zero in which case a

- 2844 single zero character is used. If a Location Reference does not exist where  $L = 12$   
 2845 leave no blank space between the two dot (.) characters.
- 2846 10. Go to Step 27.
- 2847 11. Follow the decoding procedure given in Section 3.8.1.2 (for GRAI-96) or 3.8.2.2 (for  
 2848 GRAI-170) to obtain the decimal Company Prefix  $p_1p_2...p_L$ , the decimal Asset Type  
 2849  $i_1i_2...i_{(12-L)}$ , the Filter Value  $F$ , and the Serial Number  $d_{14}d_{15}...d_K$ . If the decoding  
 2850 procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2851 12. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,  
 2852 the encoding scheme (`grai-96` or `grai-170`), a colon (:) character, the Filter  
 2853 Value  $F$  as a decimal integer, a dot (.) character, the Company Prefix  $p_1p_2...p_L$  where  
 2854 each digit (including any leading zeros) becomes the corresponding ASCII digit  
 2855 character, a dot (.) character, for  $L < 12$  the Asset Type  $i_1i_2...i_{(12-L)}$  (handled  
 2856 similarly), a dot (.) character, and the Serial Number  $d_{14}d_{15}...d_K$  as a decimal integer  
 2857 (GRAI-96) or alphanumeric character string  $s_{14}s_{15}...s_K$  (GRAI-170). For GRAI-96  
 2858 the portions corresponding to the Filter Value and Serial Number must have no  
 2859 leading zeros, except where the corresponding integer is itself zero in which case a  
 2860 single zero character is used. If an Asset Type does not exist where  $L = 12$  leave no  
 2861 blank space between the two dot (.) characters.
- 2862 13. Got to Step 27.
- 2863 14. Follow the decoding procedure given in Section 3.9.1.2 (for GIAI-96) or 3.9.2.2 (for  
 2864 GIAI-202) to obtain the decimal Company Prefix  $p_1p_2...p_L$ , the Individual Asset  
 2865 Reference  $s_{1}s_2...s_J$ , and the Filter Value  $F$ . If the decoding procedure fails, proceed  
 2866 to Step 20, otherwise proceed to the next step.
- 2867 15. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,  
 2868 the encoding scheme (`giai-96` or `giai-202`), a colon (:) character, the Filter  
 2869 Value  $F$  as a decimal integer, a dot (.) character, the Company Prefix  $p_1p_2...p_L$  where  
 2870 each digit (including any leading zeros) becomes the corresponding ASCII digit  
 2871 character, a dot (.) character, and the Individual Asset Reference  $s_{1}s_2...s_J$  (handled  
 2872 similarly). For GIAI-96 the portion corresponding to the Filter Value and the  
 2873 Individual Asset Reference must have no leading zeros, except where the  
 2874 corresponding integer is itself zero in which case a single zero character is used.
- 2875 16. Go to Step 27.
- 2876 17. Follow the decoding procedure given in Section 3.10.1.2 (for GSRN-96) to obtain the  
 2877 decimal Company Prefix  $p_1p_2...p_L$ , and the decimal Service Reference  $i_1i_2...i_{(17-L)}$ ,  
 2878 and the Filter Value  $F$ . If the decoding procedure fails, proceed to Step 20, otherwise  
 2879 proceed to the next step.
- 2880 18. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,  
 2881 the encoding scheme (`gsrn-96`), a colon (:) character, the Filter Value  $F$  as a  
 2882 decimal integer, a dot (.) character, the Company Prefix  $p_1p_2...p_L$  where each digit  
 2883 (including any leading zeros) becomes the corresponding ASCII digit character, a dot  
 2884 (.) character, and the Service Reference  $i_1i_2...i_{(17-L)}$  (handled similarly). The portion

- 2885 corresponding to the Filter Value must have no leading zeros, except where the  
 2886 corresponding integer is itself zero in which case a single zero character is used
- 2887 19. Go to Step 27.
- 2888 20. Follow the decoding procedure given in Section 3.11.1.2 (for GDTI-96) or 3.11.2.2  
 2889 (for GDTI-113) to obtain the decimal Company Prefix  $p_1p_2\dots p_L$ , the decimal  
 2890 Document Type  $i_1i_2\dots i_{(12-L)}$ , the Filter Value  $F$ , and the Serial Number  $d_{14}d_{15}\dots d_K$ . If  
 2891 the decoding procedure fails, proceed to Step 20, otherwise proceed to the next step.
- 2892 21. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`,  
 2893 the encoding scheme (`gdti-96` or `gdti-113`), a colon (`:`) character, the Filter  
 2894 Value  $F$  as a decimal integer, a dot (`.`) character, the Company Prefix  $p_1p_2\dots p_L$  where  
 2895 each digit (including any leading zeros) becomes the corresponding ASCII digit  
 2896 character, a dot (`.`) character, the Document Type  $i_1i_2\dots i_{(12-L)}$  (handled similarly), a  
 2897 dot (`.`) character, and the Serial Number  $d_{14}d_{15}\dots d_K$ . The portion corresponding to  
 2898 the Filter Value must have no leading zeros, except where the corresponding integer  
 2899 is itself zero in which case a single zero character is used.
- 2900 22. Go to Step 27.
- 2901 23. Follow the decoding procedure given in Section 3.4.1.2 to obtain the General  
 2902 Manager Number, the Object Class, and the Serial Number.
- 2903 24. Create an EPC Tag URI by concatenating the following: the string  
 2904 `urn:epc:tag:gid-96:`, the General Manager Number as a decimal number, a  
 2905 dot (`.`) character, the Object Class as a decimal number, a dot (`.`) character, and the  
 2906 Serial Number as a decimal number. Each decimal number must have no leading  
 2907 zeros, except where the integer is itself zero in which case the corresponding URI  
 2908 portion must consist of a single zero character.
- 2909 25. Go to Step 27.
- 2910 26. This tag is not a recognized EPC Tag Encoding, therefore create an EPC Raw URI by  
 2911 concatenating the following: the string `urn:epc:raw:`, the length of the bit string  
 2912 (N) expressed as a decimal integer with no leading zeros, a dot (`.`) character, a  
 2913 lowercase `x` character, and the value of the bit string considered as a single  
 2914 hexadecimal integer. The value must have a number of characters equal to the length  
 2915 (N) divided by four and rounded up to the nearest whole number, and must only use  
 2916 uppercase letters for the hexadecimal digits A, B, C, D, E, and F.
- 2917 27. The translation is now complete.
- 2918

### 2919 **5.3 Gen 2 Tag EPC Memory into EPC-URI (pure identity)**

2920 The following procedure translates the contents of the EPC Memory of a Gen 2 Tag into an  
 2921 EPC-URI:

- 2922 1. Consider bits 10x through 14x (inclusive) as a five-bit binary integer, L.

- 2923 2. Examine the “toggle” bit, bit 17x. If the toggle bit is a one, stop: this bit string  
 2924 cannot be converted into an EPC-URI. Otherwise, continue with Step 3.  
 2925 3. Extract N bits beginning with bit 20x, where N = 16L.  
 2926 4. Finish by proceeding with the procedure in Section 5.1, using the N-bit string  
 2927 extracted in Step 3.

## 2928 **5.4 Gen 2 Tag EPC Memory into Tag or Raw URI**

2929 The following procedure translates the contents of the EPC Memory of a Gen 2 Tag into  
 2930 either an EPC Tag URI or a Raw Tag URI:

- 2931 1. Consider bits 10x through 14x (inclusive) as a five-bit binary integer, L.  
 2932 2. Examine the “toggle” bit, bit 17x. If the toggle bit is a one, go to Step 5. Otherwise,  
 2933 continue with Step 3.  
 2934 3. Extract N bits beginning with bit 20x, where N = 16L.  
 2935 4. Finish by proceeding with the procedure in Section 5.2, using the N-bit string  
 2936 extracted in Step 3.  
 2937 5. This tag has an AFI, and is therefore by definition not an EPC Tag Encoding.  
 2938 Continue with the following steps.  
 2939 6. Extract bits 18x through 1Fx (inclusive) as an eight-bit binary integer, A (this is the  
 2940 AFI).  
 2941 7. Extract N bits beginning with bit 20x, where N = 16L.  
 2942 8. Create an EPC Raw URI by concatenating the following: the string  
 2943 urn:epc:raw:, the number N from Step 7 expressed as a decimal integer with no  
 2944 leading zeros, a dot (.) character, a lowercase x character, the value A from Step 6  
 2945 expressed as a two-character hexadecimal integer, a dot (.) character, a lowercase x  
 2946 character, and the value of the N-bit string from Step 7 considered as a single  
 2947 hexadecimal integer. The value must have a number of characters equal to the length  
 2948 (N) divided by four. Both the AFI and the value must only use uppercase letters for  
 2949 the hexadecimal digits A, B, C, D, E, and F.

## 2950 **5.5 URI into Bit String**

2951 The following procedure translates a URI into a bit string:

- 2952 1. If the URI is an SGTIN-URI (urn:epc:id:sgtin:), an SSCC-URI  
 2953 (urn:epc:id:sscc:), an SGLN-URI (urn:epc:id:sgln:), a GRAI-URI  
 2954 (urn:epc:id:grai:), a GIAI-URI (urn:epc:id:giai:), a GSRN-URI  
 2955 (urn:epc:id:gsrn:), a GDTI-URI (urn:epc:id:gdti:), a GID-URI  
 2956 (urn:epc:id:gid:), a DOD-URI (urn:epc:id:usdod:) or an EPC Pattern  
 2957 URI (urn:epc:pat:), the URI cannot be translated into a bit string.  
 2958 2. If the URI is a Raw Tag URI of the form urn:epc:raw:N.V, create the bit string  
 2959 by converting the second component (V) of the Raw Tag URI into a binary integer,

- 2960 whose length is equal to the first component (N) of the Raw Tag URI. If the value of  
 2961 the second component is too large to fit into a binary integer of that size, the URI  
 2962 cannot be translated into a bit string. If the URI is a Raw Tag URI of the form  
 2963  $urn:epc:raw:N.A.V$ , the URI cannot be translated into a bit string (but see the  
 2964 related procedure in Section 5.6).
- 2965 3. If the URI is an EPC Tag URI or US DoD Tag URI ( $urn:epc:tag:encName:$ ),  
 2966 parse the URI using the grammar for  $TagURI$  as given in Section 4.3.10 or for  
 2967  $DODTagURI$  as given in Section 4.3.14. If the URI cannot be parsed using these  
 2968 grammars, stop: the URI is illegal and cannot be translated into a bit string. If  
 2969  $encName$  is  $usdod-96$ , consult the appropriate U.S. Department of Defense  
 2970 document for specific translation rules. Otherwise, if  $encName$  is  $sgtin-96$  go to  
 2971 Step 4, if  $sgtin-198$  go to Step 9, if  $encName$  is  $sscc-96$  go to Step 14, if  
 2972  $encName$  is  $sgln-96$  go to Step 18 or  $sgln-195$  go to Step 23, if  $encName$  is  
 2973  $grai-96$  go to Step 28 or  $grai-170$  go to Step 33, if  $encName$  is  $giai-96$  go  
 2974 to Step 38 or  $giai-202$  go to Step 43, if  $encName$  is  $gsrn-96$  go to Step 48, if  
 2975  $encName$  is  $gdti-96$  go to Step 52, if  $gdti-113$  go to Step 56, or if  $encName$   
 2976 is  $gid-96$  go to Step 60.
- 2977 4. Let the URI be written as  
 2978  $urn:epc:tag:encName:f_1f_2...f_F.p_1p_2...p_L.i_1i_2...i_{(13-L)}.s_1s_2...s_S$ .
- 2979 5. Interpret  $f_1f_2...f_F$  as a decimal integer  $F$ .
- 2980 6. Interpret  $s_1s_2...s_S$  as a decimal integer  $S$ .
- 2981 7. Carry out the encoding procedure defined in Section 3.5.1.1 (SGTIN-96), using  
 2982  $i_1p_1p_2...p_Li_2...i_{(13-L)}0$  as the GS1 GTIN-14 (the trailing zero is a dummy check  
 2983 digit, which is ignored by the encoding procedure),  $L$  as the length of the GS1  
 2984 company prefix,  $F$  from Step 5 as the Filter Value, and  $S$  from Step 6 as the Serial  
 2985 Number. If the encoding procedure fails because an input is out of range, or because  
 2986 the procedure indicates a failure, stop: this URI cannot be encoded into a bit string.
- 2987 8. Go to Step 65.
- 2988 9. Let the URI be written as  
 2989  $urn:epc:tag:encName:f_1f_2...f_F.p_1p_2...p_L.i_1i_2...i_{(13-L)}.s_1s_2...s_S$ .
- 2990 10. Interpret  $f_1f_2...f_F$  as a decimal integer  $F$ .
- 2991 11. Interpret  $s_1s_2...s_S$  as an alphanumeric string  $S$ .
- 2992 12. Carry out the encoding procedure defined in Section 3.5.2.1 (SGTIN-198) using  
 2993  $i_1p_1p_2...p_Li_2...i_{(13-L)}0$  as the GS1 GTIN-14 (the trailing zero is a dummy check  
 2994 digit, which is ignored by the encoding procedure),  $L$  as the length of the GS1  
 2995 company prefix,  $F$  from Step 10 as the Filter Value, and  $S$  from Step 11 as the Serial  
 2996 Number. If the encoding procedure fails because an input is out of range, or because  
 2997 the procedure indicates a failure, stop: this URI cannot be encoded into a bit string.
- 2998 13. Go to Step 65.

- 2999 14. Let the URI be written as  
3000  $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(17-L)}$ .
- 3001 15. Interpret  $f_1 f_2 \dots f_F$  as a decimal integer  $F$ .
- 3002 16. Carry out the encoding procedure defined in Section 3.6.1.1 (SSCC-96), using  
3003  $i_1 p_1 p_2 \dots p_L i_2 i_3 \dots i_{(17-L)} 0$  as the GS1 SSCC (the trailing zero is a dummy check  
3004 digit, which is ignored by the encoding procedure),  $L$  as the length of the GS1  
3005 company prefix, and  $F$  from Step 15 as the Filter Value. If the encoding procedure  
3006 fails because an input is out of range, or because the procedure indicates a failure,  
3007 stop: this URI cannot be encoded into a bit string.
- 3008 17. Go to Step 65.
- 3009 18. Let the URI be written as  
3010  $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(12-L)} \cdot s_1 s_2 \dots s_S$ .
- 3011 19. Interpret  $f_1 f_2 \dots f_F$  as a decimal integer  $F$ .
- 3012 20. Interpret  $s_1 s_2 \dots s_S$  as a decimal integer  $S$ .
- 3013 21. Carry out the encoding procedure defined in Section 3.7.1.1 (SGLN-96), using  
3014  $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0$  as the GS1 GLN (the trailing zero is a dummy check digit,  
3015 which is ignored by the encoding procedure),  $L$  as the length of the GS1 company  
3016 prefix,  $F$  from Step 19 as the Filter Value, and  $S$  from Step 20 as the Extension  
3017 Component. If the encoding procedure fails because an input is out of range, or  
3018 because the procedure indicates a failure, stop: this URI cannot be encoded into a bit  
3019 string.
- 3020 22. Go to Step 65.
- 3021 23. Let the URI be written as  
3022  $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(12-L)} \cdot s_1 s_2 \dots s_S$ .
- 3023 24. Interpret  $f_1 f_2 \dots f_F$  as a decimal integer  $F$ .
- 3024 25. Interpret  $s_1 s_2 \dots s_S$  as an alphanumeric string  $S$ .
- 3025 26. Carry out the encoding procedure defined in Section 3.7.2.1 (SGLN-195), using  
3026  $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0$  as the GS1 GLN (the trailing zero is a dummy check digit,  
3027 which is ignored by the encoding procedure),  $L$  as the length of the GS1 company  
3028 prefix,  $F$  from Step 24 as the Filter Value, and  $S$  from Step 25 as the Extension  
3029 Component. If the encoding procedure fails because an input is out of range, or  
3030 because the procedure indicates a failure, stop: this URI cannot be encoded into a bit  
3031 string.
- 3032 27. Go to Step 65.
- 3033 28. Let the URI be written as  
3034  $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(12-L)} \cdot s_1 s_2 \dots s_S$ .
- 3035 29. Interpret  $f_1 f_2 \dots f_F$  as a decimal integer  $F$
- 3036 30. Interpret  $s_1 s_2 \dots s_S$  as a decimal integer  $S$ .

- 3037 31. Carry out the encoding procedure defined in Section 3.8.1.1 (GRAI-96), using  
3038  $0p_1p_2\dots p_Li_1i_2\dots i_{(12-L)}0s_1s_2\dots s_S$  as the GS1 GRAI (the second zero is a dummy  
3039 check digit, which is ignored by the encoding procedure),  $L$  as the length of the GS1  
3040 company prefix, and  $F$  from Step 29 as the Filter Value, and  $S$  from Step 30 as the  
3041 Serial Number. If the encoding procedure fails because an input is out of range, or  
3042 because the procedure indicates a failure, stop: this URI cannot be encoded into a bit  
3043 string.
- 3044 32. Go to Step 65.
- 3045 33. Let the URI be written as  
3046  $urn:epc:tag:encName:f_1f_2\dots f_F.p_1p_2\dots p_L.i_1i_2\dots i_{(12-L)}.s_1s_2\dots s_S$ .
- 3047 34. Interpret  $f_1f_2\dots f_F$  as a decimal integer  $F$ .
- 3048 35. Interpret  $s_1s_2\dots s_S$  as an alphanumeric string  $S$ .
- 3049 36. Carry out the encoding procedure defined in Section 3.8.2.1 (GRAI-170) using  
3050  $0p_1p_2\dots p_Li_1i_2\dots i_{(12-L)}0s_1s_2\dots s_S$  as the GS1 GRAI (the second zero is a dummy  
3051 check digit, which is ignored by the encoding procedure),  $L$  as the length of the GS1  
3052 company prefix, and  $F$  from Step 34 as the Filter Value, and  $S$  from Step 35 as the  
3053 Serial Number. If the encoding procedure fails because an input is out of range, or  
3054 because the procedure indicates a failure, stop: this URI cannot be encoded into a bit  
3055 string.
- 3056 37. Go to Step 65.
- 3057 38. Let the URI be written as  
3058  $urn:epc:tag:encName:f_1f_2\dots f_F.p_1p_2\dots p_L.s_1s_2\dots s_S$ .
- 3059 39. Interpret  $f_1f_2\dots f_F$  as a decimal integer  $F$
- 3060 40. Interpret  $s_1s_2\dots s_S$  as a decimal integer  $S$ .
- 3061 41. Carry out the encoding procedure defined in Section 3.9.1.1 (GIAI-96), using  
3062  $p_1p_2\dots p_Ls_1s_2\dots s_S$  as the GS1 GIAI,  $L$  as the length of the GS1 company prefix, and  
3063  $F$  from Step 39 as the Filter Value, and  $S$  from Step 40 as the Serial Number. If the  
3064 encoding procedure fails because an input is out of range, or because the procedure  
3065 indicates a failure, stop: this URI cannot be encoded into a bit string.
- 3066 42. Go to Step 65.
- 3067 43. Let the URI be written as  
3068  $urn:epc:tag:encName:f_1f_2\dots f_F.p_1p_2\dots p_L.s_1s_2\dots s_S$ .
- 3069 44. Interpret  $f_1f_2\dots f_F$  as a decimal integer  $F$ .
- 3070 45. Interpret  $s_1s_2\dots s_S$  as an alphanumeric string  $S$ .
- 3071 46. Carry out the encoding procedure defined in Section 3.9.2.1 (GIAI-202) using  
3072  $p_1p_2\dots p_Ls_1s_2\dots s_S$  as the GS1 GIAI,  $L$  as the length of the GS1 company prefix, and  
3073  $F$  from Step 44 as the Filter Value, and  $S$  from Step 45 as the Serial Number. If the  
3074 encoding procedure fails because an input is out of range, or because the procedure  
3075 indicates a failure, stop: this URI cannot be encoded into a bit string.

- 3076 47. Go to Step 65.
- 3077 48. Let the URI be written as  
3078  $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(17-L)}$ .
- 3079 49. Interpret  $f_1 f_2 \dots f_F$  as a decimal integer  $F$ .
- 3080 50. Carry out the encoding procedure defined in Section 3.10.1.1 (GSRN-96), using  
3081  $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(17-L)} 0$  as the GS1 GSRN (the trailing zero is a dummy check digit,  
3082 which is ignored by the encoding procedure),  $L$  as the length of the GS1 company  
3083 prefix, and  $F$  from Step 49 as the Filter Value. If the encoding procedure fails  
3084 because an input is out of range, or because the procedure indicates a failure, stop:  
3085 this URI cannot be encoded into a bit string.
- 3086 51. Go to Step 65.
- 3087 52. Let the URI be written as  
3088  $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(12-L)} \cdot s_1 s_2 \dots s_S$ .
- 3089 53. Interpret  $f_1 f_2 \dots f_F$  as a decimal integer  $F$ .
- 3090 54. Carry out the encoding procedure defined in Section 3.11.1.1 (GDTI-96), using  
3091  $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0 s_1 s_2 \dots s_S$  as the GS1 GDTI (the zero following  $i_{(12-L)}$  is a  
3092 dummy check digit, which is ignored by the encoding procedure),  $L$  as the length of  
3093 the GS1 company prefix and  $F$  from Step 53 as the Filter Value. If the encoding  
3094 procedure fails because an input is out of range, or because the procedure indicates a  
3095 failure, stop: this URI cannot be encoded into a bit string.
- 3096 55. Go to Step 65.
- 3097 56. Let the URI be written as  
3098  $\text{urn:epc:tag:encName} : f_1 f_2 \dots f_F \cdot p_1 p_2 \dots p_L \cdot i_1 i_2 \dots i_{(13-L)} \cdot s_1 s_2 \dots s_S$ .
- 3099 57. Interpret  $f_1 f_2 \dots f_F$  as a decimal integer  $F$ .
- 3100 58. Carry out the encoding procedure defined in Section 3.11.2.1 (GDTI-113) using  
3101  $p_1 p_2 \dots p_L i_1 i_2 \dots i_{(12-L)} 0 s_1 s_2 \dots s_S$  as the GS1 GDTI (the zero following  $i_{(12-L)}$  is a  
3102 dummy check digit, which is ignored by the encoding procedure),  $L$  as the length of  
3103 the GS1 company prefix and  $F$  from Step 57 as the Filter Value. If the encoding  
3104 procedure fails because an input is out of range, or because the procedure indicates a  
3105 failure, stop: this URI cannot be encoded into a bit string.
- 3106 59. Go to Step 65.
- 3107 60. Let the URI be written as  
3108  $\text{urn:epc:tag:encName} : m_1 m_2 \dots m_L \cdot c_1 c_2 \dots c_K \cdot s_1 s_2 \dots s_S$ .
- 3109 61. Interpret  $m_1 m_2 \dots m_L$  as a decimal integer  $M$ .
- 3110 62. Interpret  $c_1 c_2 \dots c_K$  as a decimal integer  $C$ .
- 3111 63. Interpret  $s_1 s_2 \dots s_S$  as a decimal integer  $S$ .
- 3112 64. Carry out the encoding procedure defined in Section 3.4.1.1 using  $M$  from Step 61 as  
3113 the General Manager Number,  $C$  from Step 62 as the Object Class, and  $S$  from

3114 Step 63 as the Serial Number. If the encoding procedure fails because an input is out  
3115 of range, or because the procedure indicates a failure, stop: this URI cannot be  
3116 encoded into a bit string.

3117 65. The translation is complete.

## 3118 5.6 URI into Gen 2 Tag EPC Memory

3119 The following procedure converts a URI into a sequence of bits suitable for writing into the  
3120 EPC memory of a Gen 2 Tag, starting with bit 10x (i.e., not including the CRC).

3121 1. If the URI is a Raw Tag URI of the form `urn:epc:raw:N.A.V`, calculate the  
3122 value  $L$ , where  $L = N/16$  rounded up to the nearest whole number. If  $L \geq 32$ , stop:  
3123 this URI cannot be encoded into the EPC memory of a Gen 2 Tag. If  $A \geq 256$  or if  
3124 the value  $V$  is too large to be expressed as an  $N$ -bit binary integer, stop: this URI  
3125 cannot be encoded into the EPC memory of a Gen 2 Tag. Otherwise, construct the  
3126 contents of EPC memory by concatenating the following bit strings: the value  $L$   
3127 (five bits), two zero bits (00), a single one bit (1), the value  $A$  (eight bits), and the  
3128 value  $V$  ( $16L$  bits).

3129 2. Otherwise, apply the procedure of Section 5.5 to obtain an  $N$ -bit string,  $V$ . If the  
3130 procedure of Section 5.5 fails, stop: this URI cannot be encoded into the EPC  
3131 memory of a Gen 2 Tag. Otherwise, calculate  $L = N/16$  rounded up to the nearest  
3132 whole number. Construct the contents of EPC memory by concatenating the  
3133 following bit strings: the value  $L$  (five bits), eleven zero bits (00000000000), the  
3134 value  $V$  ( $N$  bits), and as many zero bits as required to make a total of  $16(L+1)$  bits.

## 3135 6 Semantics of EPC Pattern URIs

3136 The meaning of an EPC Pattern URI (`urn:epc:pat:`) or EPC Pure Identity Pattern URI  
3137 (`urn:epc:idpat:`) can be formally defined as denoting a set of encoding-specific EPCs  
3138 or a set of pure identity EPCs, respectively.

3139 The set of EPCs denoted by a specific EPC Pattern URI is defined by the following decision  
3140 procedure, which says whether a given EPC Tag URI belongs to the set denoted by the EPC  
3141 Pattern URI.

3142 Let `urn:epc:pat:EncName:P1.P2...Pn` be an EPC Pattern URI. Let  
3143 `urn:epc:tag:EncName:C1.C2...Cn` be an EPC Tag URI, where the *EncName* field  
3144 of both URIs is the same. The number of components ( $n$ ) depends on the value of  
3145 *EncName*.

3146 First, any EPC Tag URI component  $C_i$  is said to *match* the corresponding EPC Pattern URI  
3147 component  $P_i$  if:

- 3148 •  $P_i$  is a `NumericComponent`, and  $C_i$  is equal to  $P_i$ ; or
- 3149 •  $P_i$  is a `PaddedNumericComponent`, and  $C_i$  is equal to  $P_i$  both in numeric value as  
3150 well as in length; or
- 3151 •  $P_i$  is a `GS3A3Component`, and  $C_i$  is equal to  $P_i$ , character for character; or

- 3152 •  $P_i$  is a CAGECodeOrDODAAC, and  $C_i$  is equal to  $P_i$ ; or
  - 3153 •  $P_i$  is a RangeComponent [  $l_o-h_i$  ], and  $l_o \leq C_i \leq h_i$ ; or
  - 3154 •  $P_i$  is a StarComponent (and  $C_i$  is anything at all)
- 3155 Then the EPC Tag URI is a member of the set denoted by the EPC Pattern URI if and only if  
3156  $C_i$  matches  $P_i$  for all  $1 \leq i \leq n$ .
- 3157 The set of pure identity EPCs denoted by a specific EPC Pure Identity URI is defined by a  
3158 similar decision procedure, which says whether a given EPC Pure Identity URI belongs to  
3159 the set denoted by the EPC Pure Identity Pattern URI.
- 3160 Let `urn:epc:idpat:SchemeName:P1.P2...Pn` be an EPC Pure Identity Pattern  
3161 URI. Let `urn:epc:id:SchemeName:C1.C2...Cn` be an EPC Pure Identity URI,  
3162 where the *SchemeName* field of both URIs is the same. The number of components ( $n$ )  
3163 depends on the value of *SchemeName*.
- 3164 Then the EPC Pure Identity URI is a member of the set denoted by the EPC Pure Identity  
3165 Pattern URI if and only if  $C_i$  matches  $P_i$  for all  $1 \leq i \leq n$ , where “matches” is as defined  
3166 above.

## 3167 **7 Background Information (non-normative)**

3168 This document draws from the previous work at the Auto-ID Center, and we recognize the  
3169 contribution of the following individuals: David Brock (MIT), Joe Foley (MIT), Sunny Siu  
3170 (MIT), Sanjay Sarma (MIT), and Dan Engels (MIT). In addition, we recognize the  
3171 contribution from Steve Rehling (P&G) on EPC to GTIN mapping.

3172 The following papers capture the contributions of these individuals:

- 3173 • Engels, D., Foley, J., Waldrop, J., Sarma, S. and Brock, D., "The Networked Physical  
3174 World: An Automated Identification Architecture"  
3175 2nd IEEE Workshop on Internet Applications (WIAPP '01),  
3176 <http://csdl.computer.org/comp/proceedings/wiapp/2001/1137/00/11370076.pdf>
- 3177 • Brock, David. "The Electronic Product Code (EPC), A Naming Scheme for Physical  
3178 Objects", 2001. <http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-002.pdf>
- 3179 • Brock, David. "The Compact Electronic Product Code; A 64-bit Representation of the  
3180 Electronic Product Code", 2001. <http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-008.pdf>  
3181
- 3182 • D. Engels, "The Use of the Electronic Product Code™," MIT Auto-ID Center Technical  
3183 Report MIT-AUTOID-TR009, February 2003,  
3184 <http://www.autoidlabs.org/uploads/media/MIT-AUTOID-TR009.pdf>
- 3185 • R. Moats, "URN Syntax," Internet Engineering Task Force Request for Comments RFC-  
3186 2141, May 1997, (<http://www.ietf.org/rfc/rfc2141.txt>)

3187 **8 References**

3188 [GS1 GS] “GS1 General Specifications,- Version 7.1,” January 2007, Published by GS1,  
3189 Blue Tower, Avenue Louise 326, bte10, Brussels 1009, B-1050, Belgium, [www.gs1.org](http://www.gs1.org)

3190 [MIT-TR009] D. Engels, “The Use of the Electronic Product Code™,” MIT Auto-ID Center  
3191 Technical Report MIT-AUTOID-TR009, February 2003,  
3192 <http://www.autoidlabs.org/uploads/media/MIT-AUTOID-TR009.pdf>

3193 [RFC2141] R. Moats, “URN Syntax,” Internet Engineering Task Force Request for  
3194 Comments RFC-2141, May 1997, <http://www.ietf.org/rfc/rfc2141.txt>.

3195 [DOD Constructs] “United States Department of Defense Suppliers’ Passive RFID  
3196 Information Guide,” <http://www.dodrfid.org/supplierguide.htm>

3197 [Gen2 Specification] EPCglobal “EPC Radio-Frequency Identity Protocols Class-1  
3198 Generation-2 UHF RFID Protocol for Communications at 860 MHz-960MHz Version  
3199 1.1.0,” EPCglobal Standard, October 2007,  
3200 [http://www.epcglobalinc.org/standards/uhfclg2/uhfclg2\\_1\\_1\\_0-standard-20071017.pdf](http://www.epcglobalinc.org/standards/uhfclg2/uhfclg2_1_1_0-standard-20071017.pdf)

3201

3202 **Appendix A: Encoding Scheme Summary Tables (non-**  
 3203 **normative)**

3204

<b>SGTIN Summary</b>						
<b>SGTIN-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Item Reference</b>	<b>Serial Number</b>
	8	3	3	20-40	24 - 4	38
	0011 0000 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	9,999,999 – 9 (Max .decimal range**)	274,877,906,943 (Max .decimal value)
<b>SGTIN-198</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Item Reference</b>	<b>Serial Number</b>
	8	3	3	20-40	24 - 4	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
<b>Filter Values (Non-normative)</b>		<b>SGTIN Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Indicator Digit and Item Reference</b>	
			<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digit</b>
All Others	000					
Retail Consumer Trade Item	001	0	40	12	4	1
Standard Trade Item Grouping	010	1	37	11	7	2
Single Shipping / Consumer Trade Item	011	2	34	10	10	3
Inner Trade Item Grouping not to be sold at POS	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

3205  
3206

\*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.

3207

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

3208  
3209

\*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.

<b>SGLN Summary</b>						
<b>SGLN-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Location Reference</b>	<b>Extension Component</b>
	8	3	3	20-40	21-1	41
	0011 0010 (Binary value)	(Refer to Table below for values)	(Refer to Table below for values)	999,999 – 999,999,999,999 (Max. decimal range**)	999,999 – 0 (Max. decimal range**)	2,199,023,255,551 (Max Decimal Value) 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.
<b>SGLN-195</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Location Reference</b>	<b>Extension component</b>
	8	3	3	20-40	21-1	<b>140</b>
	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
<b>Filter Values (Non-normative)</b>		<b>SGLN Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Location Reference</b>	
			<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digit</b>
All Others	000					
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

3211 \*Range of Location Reference field varies with the length of the Company Prefix

3212 \*\*Range of Company Prefix and Location Reference fields vary according to contents of the Partition field.

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

3214 \*Range of Asset Type field varies with Company Prefix.

3215 \*\*Range of Company Prefix and Asset Type fields vary according to contents of the Partition field.

3216 \*\*\* Explanation (non-normative): The Asset Type field of the GRAI-96 has four more bits than necessary given  
 3217 the capacity of that field.

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

3219 \*Range of Company Prefix and Individual Asset Reference fields vary according to contents of the Partition field.

3220 \*Range of Serial Reference field varies with the length of the Company Prefix

3221

<b>GSRN Summary</b>						
<b>GSRN-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Service Reference</b>	<b>Unallocated</b>
	8	3	3	20-40	38-18	24
	0010 1101 (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]
<b>Filter Values (Non-normative)</b>		<b>GSRN Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Service Reference</b>	
All Others	000		<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digits</b>
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

3222

3223 \*Range of Service Reference field varies with the length of the Company Prefix

3224 \*\*Range of Company Prefix and Service Reference fields vary according to the contents of the Partition field.

<b>GDTI Summary</b>						
<b>GDTI-96</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Document Type</b>	<b>Serial Number</b>
	8	3	3	20-40	21 – 1	41
	0010 1100 (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)
<b>GDTI-113</b>	<b>Header</b>	<b>Filter Value</b>	<b>Partition</b>	<b>Company Prefix</b>	<b>Document Type</b>	<b>Serial Number</b>
	8	3	3	20-40	21 – 1	58
	0011 1010 (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 17 numeric characters
<b>Filter Values (Non-normative)</b>		<b>GDTI Partition Table</b>				
<b>Type</b>	<b>Binary Value</b>	<b>Partition Value</b>	<b>Company Prefix</b>		<b>Document Type</b>	
			<b>Bits</b>	<b>Digits</b>	<b>Bits</b>	<b>Digit</b>
All Others	000					
Reserved	001	0	40	12	1	0
Reserved	010	1	37	11	4	1
Reserved	011	2	34	10	7	2
Reserved	100	3	30	9	11	3
Reserved	101	4	27	8	14	4
Reserved	110	5	24	7	17	5
Reserved	111	6	20	6	21	6

3226 \*Range of Document Type field varies with Company Prefix.

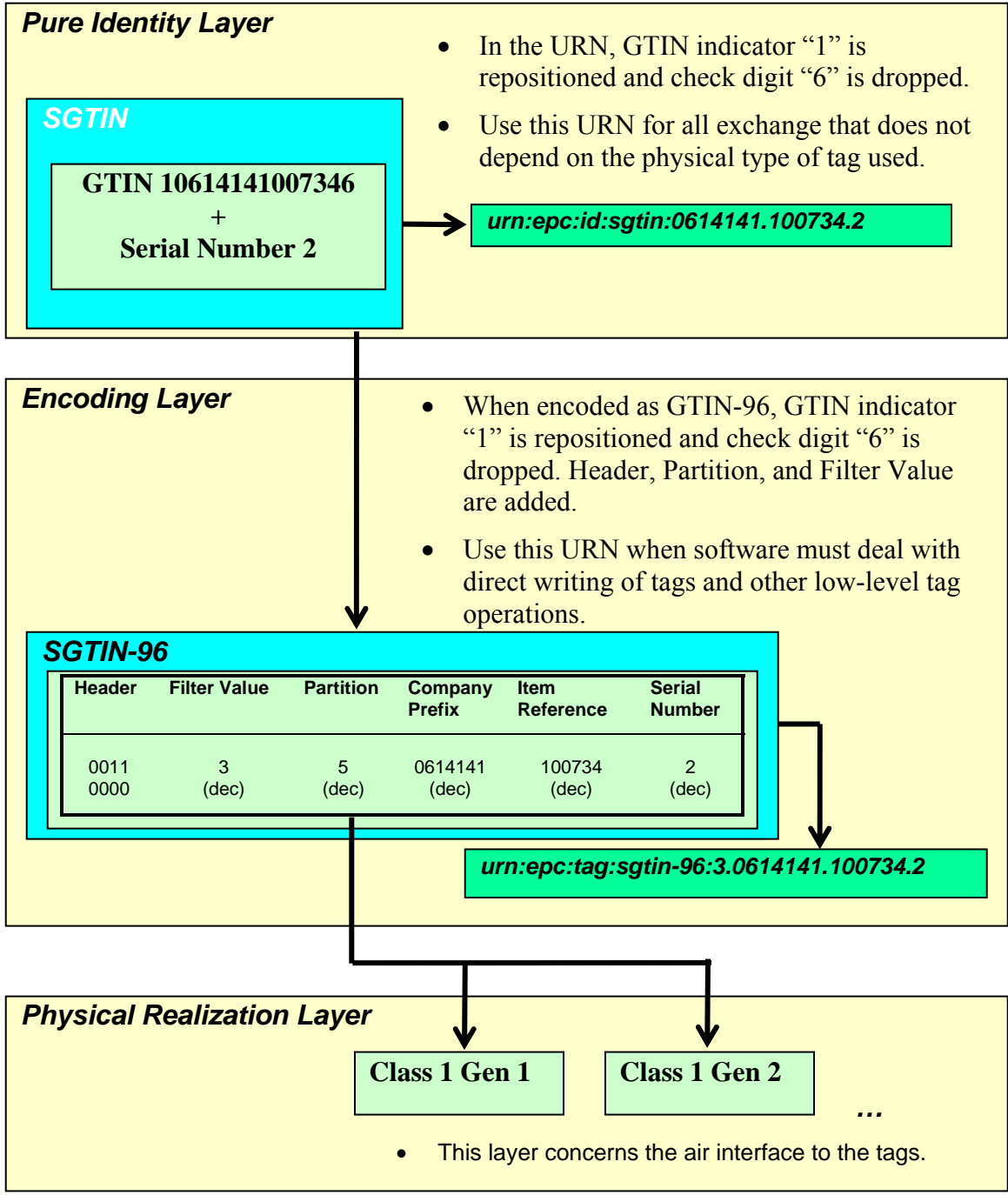
3227 \*\*Range of Company Prefix and Document Type fields vary according to contents of the Partition field.

3228

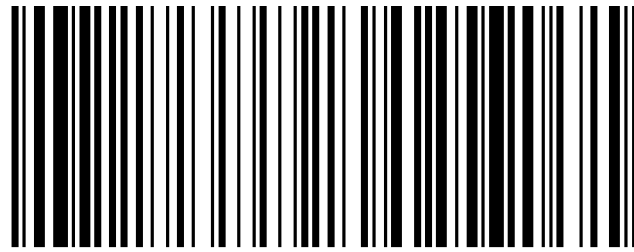
3229

3230 **Appendix B: Example of a Specific Trade Item <SGTIN>**  
 3231 **(non-normative)**

3232 This section presents an example of a specific trade item using SGTIN (Serialized GTIN).  
 3233 Each representation serves a distinct purpose in the software stack. Generally, the highest  
 3234 applicable level should be used. The GTIN used in the example is 10614141007346.



3235



3236

3237

3238

3239

	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)

3240

3241

3242

3243

3244

3245

3246

3247

3248

3249

3250

3251

3252

Explanation of SGTIN Filter Values (non-normative).

3253

3254

3255

3256

3257

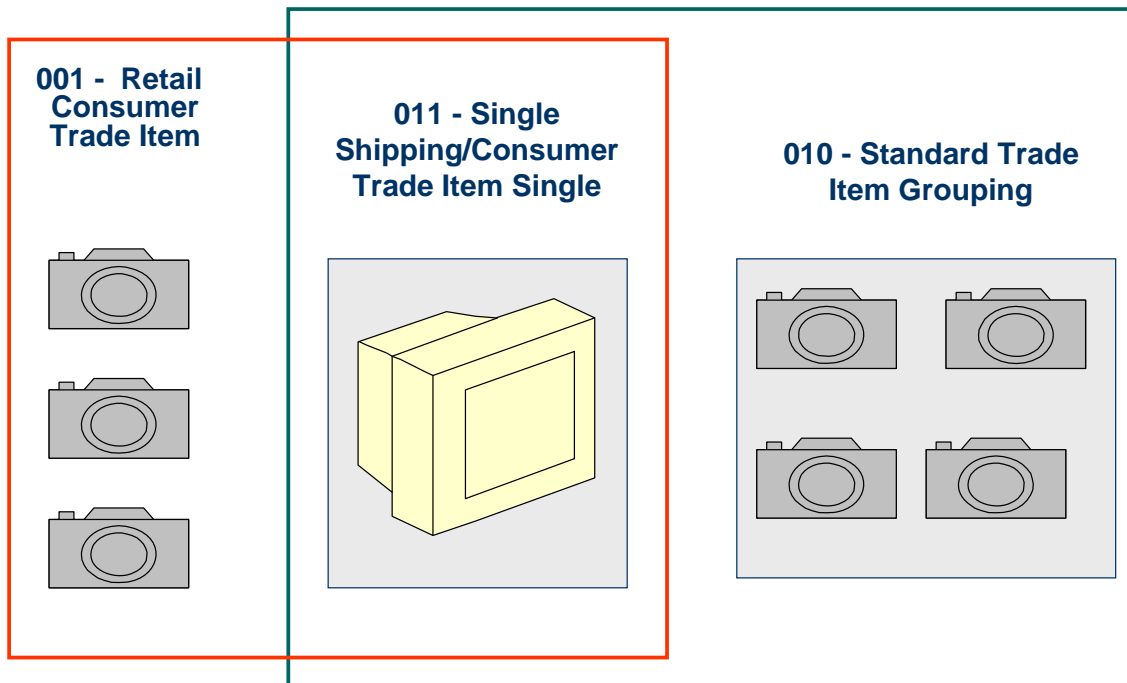
SGTINs can be assigned at several levels, including: item, inner pack, case, and pallet. RFID can read through cardboard, and reading un-needed tags can slow us down, so Filter Values are used to “filter in” desired tags, or “filter out” unwanted tags. Filter values are used within the key type (i.e.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

- (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.

3258 for now until the community gains more practical experience in their use. Therefore the  
3259 three major categories of SGTIN filter values can be thought of in the following high level  
3260 terms:

- 3261 • Single Unit: A Retail Consumer Trade Item
- 3262 • Not-a-single unit: A Standard Trade Item Grouping
- 3263 • Items that could be included in both categories: For example, a Single Shipping  
3264 container that contains a Single Consumer Trade Item
- 3265

## Three Filter Values



3266

3267

3268 **Appendix C: Decimal values of powers of 2 Table (non-**  
3269 **normative)**

3270

n	(2^n) <sub>10</sub>	n	(2^n) <sub>10</sub>
0	1	33	8,589,934,592
1	2	34	17,179,869,184
2	4	35	34,359,738,368
3	8	36	68,719,476,736
4	16	37	137,438,953,472
5	32	38	274,877,906,944
6	64	39	549,755,813,888
7	128	40	1,099,511,627,776
8	256	41	2,199,023,255,552
9	512	42	4,398,046,511,104
10	1,024	43	8,796,093,022,208
11	2,048	44	17,592,186,044,416
12	4,096	45	35,184,372,088,832
13	8,192	46	70,368,744,177,664
14	16,384	47	140,737,488,355,328
15	32,768	48	281,474,976,710,656
16	65,536	49	562,949,953,421,312
17	131,072	50	1,125,899,906,842,624
18	262,144	51	2,251,799,813,685,248
19	524,288	52	4,503,599,627,370,496
20	1,048,576	53	9,007,199,254,740,992
21	2,097,152	54	18,014,398,509,481,984
22	4,194,304	55	36,028,797,018,963,968
23	8,388,608	56	72,057,594,037,927,936
24	16,777,216	57	144,115,188,075,855,872
25	33,554,432	58	288,230,376,151,711,744
26	67,108,864	59	576,460,752,303,423,488
27	143,217,728	60	1,152,921,504,606,846,976
28	286,435,456	61	2,305,843,009,213,693,952
29	572,870,912	62	4,611,686,018,427,387,904
30	1,145,741,824	63	9,223,372,036,854,775,808
31	2,291,483,648	64	18,446,744,073,709,551,616
32	4,582,967,296		

3271

3272 **Appendix D: List of Abbreviations**

3273

BAG	Business Action Group
EPC	Electronic Product Code
EPCIS	EPC Information Services
GDTI	Global Document Type Identifier
GIAI	Global Individual Asset Identifier
GID	General Identifier
GLN	Global Location Number
GRAI	Global Returnable Asset Identifier
GSRN	Global Service Relation Number
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

3274

3275

3276 **Appendix E: GS1 General Specifications Version 7.1 (non-**  
3277 **normative)**

3278 (Section 3 Definition of Element Strings and Section 3.7 EPCglobal Tag Data Standard.)

3279 This section provides GS1 approval of this version of the EPCglobal® Tag Data Standard  
3280 with the following GS1 Application Identifier definition restrictions:

3281 Companies should use the GS1 specifications to define the applicable fields in databases and  
3282 other ICT-systems.

3283 For GS1 use of EPC96-bit tags, the following applies:

- 3284 • **AI (00) SSCC** (no restrictions)
- 3285 • **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is  
3286 restricted to permit assignment of 274,877,906,943 numeric-only serial numbers)
- 3287 • **AI (414) GLN + AI (254) GLN Extension Component:** The Tag Data Standard V1.1 R1.27  
3288 is approved for the use of GLN Extension with the restrictions specified in Section 2.4.6.1 of  
3289 the GS1 General Specifications..
- 3290 • **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier  
3291 definition is restricted to permit assignment of 274,877,906,943 numeric-only serial numbers  
3292 and the serial number element is mandatory.
- 3293 • **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier  
3294 definition is restricted to permit assignment of 4,611,686,018,427,387,904 numeric-only serial  
3295 numbers.

3296 For GS1 use of EPC longer then 96-bit tags, the following applies:

- 3297 • **AI (00) SSCC** (no restrictions)
- 3298 • **AI (01) GTIN + AI (21) Serial Number:** (no restrictions)
- 3299 • **AI (414) GLN + AI (254) Extension Component:** (no restrictions).
- 3300 • **AI (8003) GRAI Serial Number:** (no restrictions)
- 3301 • **AI (8004) GIAI Serial Number:** (no restrictions)

## Appendix F: GS1 Alphanumeric Character Set (Normative)

ISO/IEC 646 Subset

<b>Unique Graphic Character Allocations</b>					
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

3304 Notes

3305 Readers should be aware that this table is derived from [GS1 GS] and may include  
 3306 discrepancy with the original specification at any given time. Readers are advised to always  
 3307 consult the original specification upon implementation.

3308 This table specifies the allowed subset of ISO/IEC 646 characters that shall be used for  
 3309 encoding alphanumeric Serial Number/Extension Component in this standard. The SGTIN-  
 3310 198, SGLN-195, GRAI-170 and GIAI-202 encodings use this table.

3311 Each entry in this table gives a 7-bit code for a character, expressed in hexadecimal. For  
 3312 example, “Capital Letter K” has a 7-bit code of 1001011, expressed as “4B” in the table.

3313 The 7-bit codes in this table are identical to ISO/IEC 646 (ASCII) character codes.

3314

3315 **Appendix G: Acknowledgement of Contributors and**  
 3316 **Companies Opted-in during the Creation of this Standard**  
 3317 **(Informative)**

3318

3319 Disclaimer

3320 *Whilst every effort has been made to ensure that this document and the information*  
 3321 *contained herein are correct, EPCglobal and any other party involved in the creation*  
 3322 *of the document hereby state that the document is provided on an “as is” basis*  
 3323 *without warranty, either expressed or implied, including but not limited to any*  
 3324 *warranty that the use of the information herein with not infringe any rights, of*  
 3325 *accuracy or fitness for purpose, and hereby disclaim any liability, direct or indirect,*  
 3326 *for damages or loss relating to the use of the document.*

3327

3328 Below is a list of active participants and contributors in the development of TDS 1.4.  
 3329 This list does not acknowledge those who only monitored the process or those who  
 3330 chose not to have their name listed here. Active participants status was granted to  
 3331 those who generated emails, attended face-to-face meetings and conference calls  
 3332 that were associated with the development of this Standard.  
 3333

Mark	Frey	EPCglobal Inc.	SAG Facilitator
Sylvia	Stein	GS1 Netherlands (EAN.nl)	WG Facilitator
Vijay	Sundhar	Ahold NV	WG co-Chair
John	Anderla	Kimberly-Clark	Editor
Rick	Schuessler	Symbol Technologies Inc, a Motorola Co.	WG co-Chair
Richard	Bach	GlobeRanger	
Theron	Stanford	Impinj	
Ken	Traub	Ken Traub Consulting LLC	
Mark	Harrison	Auto-ID Labs - Cambridge	
Scott	Barvick	Reva Systems	
Sprague	Ackley	Intermec Technologies Corporation	
John	Kessler	Avery Dennison	
Craig Alan	Repec	GS1 Germany (CCG)	
Gay	Whitney	EPCglobal Inc.	
Craig	Harmon	Q.E.D. Systems	
Rob	Buck	Intermec Technologies Corporation	
Clive	Hohberger	Zebra Technologies Corporation	
Stephen	Miles	Auto-ID Labs - MIT	
Shigeya	Suzuki	Auto-ID Labs - Japan	
Bud	Biswas	Polaris Networks	
Timo	Liu	Regal Scan Tech	
Kay	Labinsky	Sandlab Corp.	
Larry	Moore	TEGO, Inc.	
Margaret	Wasserman	ThingMagic, LLC	

3334  
 3335  
 3336  
 3337  
 3338  
 3339  
 3340  
 3341

The following list in corporate alphabetical order contains all companies that were opted-in to the Tag Data and Translation Standard Working Group and have signed the EPCglobal IP Policy.

<b>Company</b>
Acer Cybercenter Service Inc.
Ahold NV
Allixon Co., Ltd
Altria Group, Inc./Kraft Foods
AMCO TEC International Inc.

AMOS Technologies Inc.
AMOS Technologies Inc.
Applied Wireless (AWID)
Atmel GmbH
Auto-ID Labs - ADE
Auto-ID Labs - Cambridge
Auto-ID Labs - Fudan University
Auto-ID Labs - ICU
Auto-ID Labs - Japan
Auto-ID Labs - MIT
Auto-ID Labs - University of St Gallen
AXWAY/formerly Cyclone
<a href="#">Avery Dennison</a>
BEA Systems
Benedicta
Cheng-Loong Corporation
Cognizant Technology Solutions
EB (Formerly 7iD)
ECO, Inc.
EPCglobal Inc.
ETRI - Electronics & Telecommunication Research Institute
France Telecom
GlaxoSmithKline
GlobeRanger
GS1 Australia EAN
GS1 China
GS1 Germany (CCG)
GS1 Hong Kong
GS1 International
GS1 Japan
GS1 Netherlands (EAN.nl)
GS1 South Korea
GS1 Sweden AB (EAN)
GS1 Taiwan (EAN)
GS1 US
iControl, Inc.
Impinj
Innovision Res & Techno
Intelleflex
Intermec Technologies Corporation
Johnson & Johnson
Ken Traub Consulting LLC
Kimberly-Clark
KL-NET
KTNET - KOREA TRADE NETWORK
Kun Shan University Information Engineering Department
LIT (Research Ctr for Logistics Info Tech)

Lockheed Martin - Savi Technology Divison
Lockheed Martin, Corp.
Manhattan Associates
MetaBiz
Microelectronics Technology, Inc.
MITSUI & CO., LTD.
NEC Corporation
Nestle
NXP Semiconductors
Oracle Corporation
Paxar
Polaris Networks
Printronic
Procter & Gamble Company
Q.E.D. Systems
Regal Scan Tech
RetailTech
Reva Systems
RF-IT Solutions GmbH
RFID Research Center, Chang Jung Christian University
Sandlab Corp.
Sandlinks
Schering-Plough Corp.
Secure RF
STMicroelectronics
Symbol Technologies Inc, a Motorola Co.
Tagent Corporation
Target Corporation
TEGO, Inc.
Tesco
ThingMagic, LLC
Tibco Software, Inc
Toppan Printing Co., Ltd
Toray International, Inc.
TrueDemand Software
US Defense Logistics Agency (DoD)
Ussen Limited Company
VeriSign
WAL-MART STORES, INC.
Yuen Foong Yu Paper
Zebra Technologies Corporation

3342

3343