# DECLARATION OF JUNE ANN MUNFORD

1. My name is June Ann Munford. I am over the age of 18, have personal knowledge of the facts set forth herein, and am competent to testify to the same.

2. I earned a Master of Library and Information Science (MLIS) from the University of Wisconsin-Milwaukee in 2009. I have over ten years of experience in the library/information science field. Beginning in 2004, I have served in various positions in the public library sector including Assistant Librarian, Youth Services Librarian and Library Director. I have attached my Curriculum Vitae as Appendix CV.

3. During my career in the library profession, I have been responsible for materials acquisition for multiple libraries. In that position, I have cataloged, purchased and processed incoming library works. That includes purchasing materials directly from vendors, recording publishing data from the material in question, creating detailed material records for library catalogs and physically preparing that material for circulation. In addition to my experience in acquisitions, I was also responsible for analyzing large collections of library materials, tailoring library records for optimal catalog

search performance and creating lending agreements between libraries during my time as a Library Director.

4. I am fully familiar with the catalog record creation process in the library sector. In preparing a material for public availability, a library catalog record describing that material would be created. These records are typically written in Machine Readable Catalog (herein referred to as "MARC") code and contain information such as a physical description of the material, metadata from the material's publisher, and date of library acquisition. In particular, the 008 field of the MARC record is reserved for denoting the date of creation of the library record itself. As this typically occurs during the process of preparing materials for public access, it is my experience that an item's MARC record indicates the date of an item's public availability.

5. Typically, in creating a MARC record, a librarian would gather various bits of metadata such as book title, publisher and subject headings among others and assign each value to a relevant numerical field. For example, a book's physical description is tracked in field 300 while title/attribution is tracked in field 245. The 008 field of the MARC record is reserved for denoting the creation of the library record itself. As this is the only date reflecting the inclusion of said materials within the library's collection, it is my experience

that an item's 008 field accurately indicates the date of an item's public availability.

6. I am familiar with the Internet Archive, a digital library formally certified by the State of California as a public library. Among other services that the Internet Archive makes available to the general public is the Wayback Machine, an online archive. The Internet Archive's Wayback Machine service archives webpages as of a certain capture date to track changes in the web over time. The Internet Archive has been in operation as a nonprofit library since 1996 and has hosted the Wayback Machine service since its inception in 2001. During my time as a librarian, I frequently used the Internet Archive's Wayback Machine for research and instruction purposes. This includes teaching instructional classes on using the Wayback Machine to library patrons and using the Wayback Machine to research reference inquiries that require hard-to-find online resources. I consider the Internet Archive's recordskeeping to be as rigorous and detailed as other formal library recordskeeping practices such as MARC records, OCLC records and Dublin Core.

7. I have reviewed Exhibit EX-1009, *RFID for Dummies* by Patrick J. Sweeney.

8. Attached hereto as Appendix SWEENEY01 is a true and correct copy of the MARC record for *RFID for Dummies* as held by the University of Akron library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix SWEENEY01 accurately describes the title, author, publisher, and ISBN number of *RFID for Dummies* by Patrick J. Sweeney.

9. Attached hereto as Appendix SWEENEY02 is a true and correct copy of selections from *RFID for Dummies* as held by the University of Akron library. I secured these scans myself. In comparing Exhibit EX-1009 to Appendix SWEENEY02, it is my determination that Exhibit EX-1009 is a true and correct copy of *RFID for Dummies* by Patrick J. Sweeney.

10. The 008 field of the MARC record in Appendix SWEENEY01 indicates the date of record creation. The 008 field of Appendix SWEENEY01 indicates the University of Akron library first acquired *RFID for Dummies* as of February 1, 2005. Considering this information, it is my determination that *RFID for Dummies* was made available to the public shortly after its initial acquisition as of February 1, 2005.

11. I have reviewed Exhibit EX-1011, *EPC Tag Data Standards Version 1.1 Rev.1.24* by EPCglobal.

12. Attached hereto as Appendix EPC01 are screen captures of the Internet Archive Wayback Machine record for a page entitled "Version 1.0 Specifications" as hosted on the EPCglobal website. This record for the URL http://epcglobalinc.org:80/standards_technology/specifications.html can be found at https://web.archive.org/web/20040612103254/http://epcglobalinc.org:80/standards_technology/specifications.html. This page was captured by the Internet Archive as of June 12, 2004. I secured these screen captures myself.

13. Appendix EPC01 is a web page describing various EPC specifications, including *EPC Tag Data Standards Version 1.1 Rev.1.24*. Page 2 of this Appendix includes a link titled 'EPC Tag Data Standards Version 1.1' that directs to a PDF copy of *EPC Tag Data Standards Version 1.1 Rev.1.24*.

14. Attached hereto as Appendix EPC02 are screen captures of the Internet Archive Wayback Machine record for a page entitled "EPCTM Tag Data Standards Version 1.1 Rev.1.24" as hosted on the EPCglobal website. This record for the URL

http://www.epcglobalinc.org:80/standards_technology/EPCTagDataSpecification11rev124.pdf can be found at https://web.archive.org/web/20040612222224/http://www.epcglobalinc.org:80/standards_technology/EPCTagDataSpecification11rev124.pdf. This page was captured by the Internet Archive as of June 12, 2004. I secured these screen captures myself.

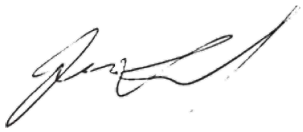
15. Appendix EPC02 is a website presenting a complete PDF copy of *EPC Tag Data Standards Version 1.1 Rev.1.24.* In comparing Exhibit EX-1011 to Appendix EPC02, it is my determination that Exhibit EX-1011 is a true and correct copy of *EPC Tag Data Standards Version 1.1 Rev.1.24*. Given that the record preserved in Appendix EPC02 was first captured as of June 12, 2004, it is my determination that *EPC Tag Data Standards Version 1.1 Rev.1.24* was first made available to the public as of June 12, 2004 if not earlier via the original EPCglobal website.

16. I have been retained on behalf of the Petitioner to provide assistance in the above-illustrated matter in establishing the authenticity and public availability of the documents discussed in this declaration. I am being compensated for my services in this matter at the rate of $200.00 per hour

plus reasonable expenses. My statements are objective, and my compensation does not depend on the outcome of this matter.

17. I declare under penalty of perjury that the foregoing is true and correct. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: 9/10/2024



June Ann Munford

# Exhibit CV

June A. Munford
Curriculum Vitae

## Education

University of Wisconsin-Milwaukee - MS, Library & Information Science, 2009
Milwaukee, WI

- Coursework included cataloging, metadata, data analysis, library systems, management strategies and collection development.
- Specialized in library advocacy, cataloging and public administration.

Grand Valley State University - BA, English Language & Literature, 2008
Allendale, MI

- Coursework included linguistics, documentation and literary analysis.
- Minor in political science with a focus in local-level economics and government.

## Professional Experience

Researcher / Expert Witness, October 2017 – present
Freelance ● Pittsburgh, Pennsylvania & Grand Rapids, Michigan

- Material authentication and public accessibility determination. Declarations of authenticity and/or public accessibility provided upon research completion. Experienced with appeals and deposition process.

- Research provided on topics of public library operations, material publication history, digital database services and legacy web resources.

- Past clients include Alston & Bird, Arnold & Porter, Baker Botts, Fish & Richardson, Erise IP, Irell & Manella, O'Melveny & Myers, Perkins-Coie, Pillsbury Winthrop Shaw Pittman and Slayden Grubert Beard.

Library Director, February 2013 - March 2015
Dowagiac District Library ● Dowagiac, Michigan

- Executive administrator of the Dowagiac District Library. Located in

Southwest Michigan, this library has a service area of 13,000, an annual operating budget of over $400,000 and total assets of approximately $1,300,000.

● Developed careful budgeting guidelines to produce a 15% surplus during the 2013-2014 & 2014-2015 fiscal years while being audited.

● Using this budget surplus, oversaw significant library investments including the purchase of property for a future building site, demolition of existing buildings and building renovation projects on the current facility.

● Led the organization and digitization of the library's archival records.

● Served as the public representative for the library, developing business relationships with local school, museum and tribal government entities.

● Developed an objective-based analysis system for measuring library services - including a full collection analysis of the library's 50,000+ circulating items and their records.

November 2010 - January 2013
Librarian & Branch Manager, Anchorage Public Library ● Anchorage, Alaska

● Headed the 2013 Anchorage Reads community reading campaign including event planning, staging public performances and creating marketing materials for mass distribution.

● Co-led the social media department of the library's marketing team, drafting social media guidelines, creating original content and instituting long-term planning via content calendars.

● Developed business relationships with The Boys & Girls Club, Anchorage School District and the US Army to establish summer reading programs for children.

June 2004 - September 2005, September 2006 - October 2013
Library Assistant, Hart Area Public Library
Hart, MI

● Responsible for verifying imported MARC records and original MARC

cataloging for the local-level collection as well as the Michigan Electronic Library.

● Handled OCLC Worldcat interlibrary loan requests & fulfillment via ongoing communication with lending libraries.

**Professional Involvement**

Alaska Library Association - Anchorage Chapter
● Treasurer, 2012

Library Of Michigan
● Level VII Certification, 2008
● Level II Certification, 2013

Michigan Library Association Annual Conference 2014
● New Directors Conference Panel Member

Southwest Michigan Library Cooperative
● Represented the Dowagiac District Library, 2013-2015

**Professional Development**

Library Of Michigan Beginning Workshop, May 2008
Petoskey, MI
● Received training in cataloging, local history, collection management, children's literacy and reference service.

Public Library Association Intensive Library Management Training, October 2011
Nashville, TN
● Attended a five-day workshop focused on strategic planning, staff management, statistical analysis, collections and cataloging theory.

Alaska Library Association Annual Conference 2012 - Fairbanks, February 2012
Fairbanks, AK
● Attended seminars on EBSCO advanced search methods, budgeting, cataloging, database usage and marketing.

**Depositions**

2019 ● Fish & Richardson

Apple v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282 and 39521-00421IP2)

2019 ● Erise IP

Implicit, LLC v. Netscout Systems, Inc (Civil Action No. 2:18-cv-53-JRG)

2019 ● Perkins-Coie

Adobe Inc. v. RAH Color Technologies LLC (Cases IPR2019-00627, IPR2019-00628, IPR2019-00629 and IPR2019-00646)

2020 ● O'Melveny & Myers

Maxell, Ltd. v. Apple Inc. (Case No. 5:19-cv-00036-RWS)

2021 ● Pillsbury Winthrop Shaw Pittman LLP

Intel v. SRC (IPR2020-1449)

2022 ● Perkins-Coie

Realtek v. Future Link (IPR2021-01182)

2023 ● Fish & Richardson

Neuroderm Ltd. v. Abbvie, Inc (Case No. PGR2022-00040)

2023 ● Fish & Richardson

Nearmap US Inc. v. Pictometry International Corp. (IPR2022-00735)

2023 ● Fish & Richardson

Samsung Electronics v. MemoryWeb LLC (Case No. 39843-0136PS1)

2023 ● Pillsbury Winthrop Shaw Whitman LLP

Gravel Rating Systems v. Costco Wholesale Corp. (Civil Action No. 4:21-cv-149-ALM)

2024 ● Willkie-Farr

Netflix, Inc. v. VideoLabs. Inc. (IPR2023-00628)

**Limited Case and Clientele History**

Alston & Bird

- Ericsson

  v. Collision Communications (IPR2022-01233)

- Nokia

  v. Neptune Subsea, Xtera (Case No. 1:17-cv-01876)

- Universal Electronics Inc

  v. Roku Inc (IPR2022-00818)

Arnold & Porter

- Ivantis

  v. Glaukos (Case No. 8:18-cv-00620)

- Samsung

  v. Jawbone (Case No. 2:21-cv-00186)

Benesch Friedlander Coplan & Aronoff

- Voyis

  v. Cathx (Case No.  5:21-cv-00077-RWS)

Deschert LLP

- Smaxtec Animal Care

  v. ST Reproductive Technologies, LLC (IPR2024-00885)

Erise I.P.

- Apple

  v. Ericsson Inc. (IPR2022-00715)

  v. Future Link Systems (IPRs 6317804, 6622108, 6807505, and 7917680)

  v. INVT (Case No. 20-1881)

  v. Navblazer LLC (IPR2020-01253)

  v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282, 39521-00421IP2)

  v. Quest Nettech Corp (Case No. 2:19-cv-00118-JRG)

  v. Telefonaktiebolaget LM Ericsson (IPR2022-00275)

  v. Theta IP, LLC (IPR2024-00818)

- Fanduel

  v. CGT (Case No. 19-1393)

- Garmin

  v. Phillips North America LLC (Case No. 2:19-cv-6301-AB-KS)

- Netscout
    - v. Longhorn HD LLC (Case No. 2:20-cv-00349)
    - v. Implicit, LLC (Civil Action No. 2:18-cv-53-JRG)

- Sony Interactive Entertainment LLC
    - v. Bot M8 LLC (IPR2020-01288)
    - v. Infernal Technology LLC (Case No. 2:19-CV-00248-JRG)

- Unified Patents
    - v. GE Video Compression (Civil Action No. 2:19-cv-248)

Fish & Richardson

- Apple
    - v. AliveCor (3:21-cv-03958)
    - v. LBS Innovations (Case No. 2:19-cv-00119-JRG-RSP)
    - v. Koss Corporation (IPR2021-00305)
    - v. Masimo (IPR 50095-0012IP1, 50095-0012IP2, 50095-0013IP1, 50095-0013IP2, 50095-0006IP1, 50095-0135IP1)
    - v. Neonode (Case No. 21-cv-08872-EMC)
    - v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282, 39521-00421IP2)

- Dell
    - v. Neo Wireless (IPR2022-00616)

- Dish Network
    - v. Entropic Communications, LLC (Case No. 2:2023-CV-01043)

v. Realtime Adaptive Streaming (Case No. 1:17-CV-02097-RBJ)

v. TQ Delta LLC (Case No. 18-1798)

- Evapco Dry Cooling

  v. SPG Dry Cooling (IPR2021-00688)

- Genetec

  v. Sensormatic Electronics (Case No. 1:20-CV-00760)

- Huawei

  v. Bell Northern Research LLC (IPR2019-01174)

- Kianxis

  v. Blue Yonder (Case No. 3:20-cv-03636)

- LG Electronics

  v. Bell Northern Research LLC (Case No. 3:18-cv-2864-CAB-BLM)

- Metaswitch

  v. Sonus Networks (IPR2018-01719)

- Microsoft

  v. Throughputer Inc (IPR2022-00757)

- Mom Enterprises

  v. Ddrops Company (Case No. 1:22-cv-00332)

- MLC Intellectual Property

v. MicronTech (Case No. 3:14-cv-03657-SI)

- Nearmap Inc

  v. EagleView Technologies (IPR2022-01009)

- Neuroderm Ltd.

  v. Abbvie, Inc (Case No. PGR2022-00040)

- Realtek Semiconductor

  v. Future Link (IPR2021-01182)

- Quectel

  v. Koninklijke Philips (Case No. 1:20-cv-01710)

- Samsung

  v. Aire Technology (IPR2022-00877)

  v. Bell Northern Research (Case No. 2:19-cv-00286-JRG)

  v. Communication Technologies Inc (IPR2022-01221)

  v. Jawbone Innovations (IPR2022-00865)

  v. MemoryWeb LLC (IPR2022-00885)

  v. Dodots Licensing Solutions (IPR2023-00701)

  v. Telefonaktiebolaget LM Ericsson (IPR2021-00615)

- Texas Instruments

  v. Vantage Micro LLC (IPR2020-01261)

- Xilinx

  v. Sentient Sensors LCC (Case No. 1:22-cv-00173)

Groombridge Wu

- Nearmap

  v. Eagle View Technologies Inc. and Pictometry International Corp. (Case No. 2:21-cv-00283-TS-DAO)

Irell & Manella

- Curium

O'Melveny & Myers

- Apple

  v. Maxell (Case No. 5:19-cv-00036-RWS)

- Disney Media & Entertainment Distribution LLC

  v. Digital Media Technology Holdings, LLC (Case No. 2:22-cv-01642)

- Micron Technology Inc.

  v. Besang Inc (IPR2023-00900)

- Samsung

  v. Daedalus Prime LLC (1335 Investigation)

Perkins-Coie

- Amazon

  v. Nokia (IPR2024-00847 and IPR2024-00848)

- Heru Industries

  v. The UAB Research Foundation (IPR2022-01148)

- Intel Corporation

  v. BeSang Inc. (IPR2023-00991)

  v. BitMICRO (Case No. 5:23-cv-00625)

  v. Telefonaktiebolaget L M Ericsson (IPR2024-00610)

- Realtek Semiconductor

  v. Future Link (IPR2021-01182)

- Twitter Inc

  v. VOIP-Pal.com (Case No. 3:20-cv-02397-JD)

- TCL Industries

  v. Koninklijke Philips NV (IPR2021-00495, IPR2021-00496 and IPR2021-00497)

Pillsbury Winthrop Shaw Pittman

- Intel

  v. FG SRC LLC (Case No. 6:20-cv-00315)

- Gravel Rating Systems

  v. Costco (Case No. 4:21-cv-149)
  v. Lowe's Home Centers (Case No. 4:21-cv-150)
  v. T-Mobile USA (Case No. 4:21-cv-152)
  v. Kohl's Inc. (Case No. 4:21-cv-258)
  v. Under Armor (Case No. 4:21-cv-356)

Quinn Emanuel

- ServiceNow, Inc.

  v. InQuisient, Inc. (Case No. 22-900-CJB)

- Wolfspeed

  v. The Trustees of Purdue University (Case No. 1:2021-cv-00840)

Sheppard, Mullin, Richter & Hampton LLP

- Cadence Design Systems Inc.

  v. Semiconductor Design Technologies (Case No. 3:23-cv-01001)

Stern, Kessler, Goldstein & Fox

- Preliminary Research

Willkie, Farr & Gallagher LLP

- Neurent

  v. The Foundry, LLC (IPR2024-00669)

- Netflix, Inc.

  v. VideoLabs, Inc. (IPR2023-00628)

# Exhibit EPC01

DONATE

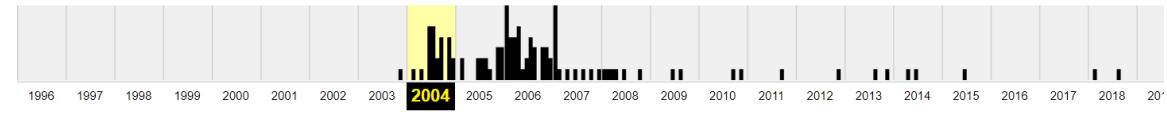
INTERNET ARCHIVE WayBack Machine

Explore more than 866 billion web pages saved over time

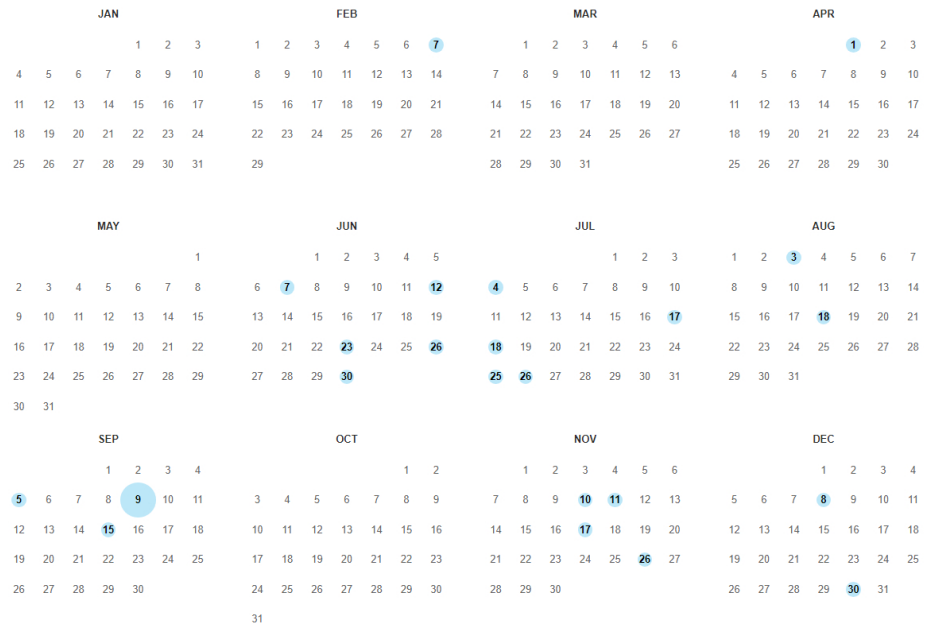
http://epcglobalinc.org:80/standards_technology/specifications.html

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **110 times** between November 7, 2003 and August 1, 2018.




Sat, 12 Jun 2004 10:32:54 GMT (why: alexa_dx, alexacrawls)




## Note

This calendar view maps the number of times **http://epcglobalinc.org:80/standards_technology/specifications.html** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use.

https://web.archive.org/web/20040612103254/http://epcglobalinc.org:80/standards_technology/specifications.html 2024-08-26T13:41:44-04:00

INTERNET ARCHIVE WayBackMachine
http://epcglobalinc.org/standards_technology/specifications.html Go
110 captures
7 Nov 2003 - 1 Aug 2018
APR JUN 12 JUL
2003 2004 2005
About this capture





## Version 1.0 Specifications

Specifications provide the overall system definition and how functional requirements are partitioned across various subsystems. They signify the output that advances a community toward a common objective, namely, to create an Electronic Product Code (EPC) Network that End Users can examine for their supply chain implementations.

The set of specifications that comprise Version 1.0 provide many of the building blocks required to implement an EPCglobal Network. The Version 1.0 Specifications are at various stages of completion. Some have reached recommendation status, where final approval is imminent. Others are in working draft form where additional definition is required. We would encourage you to examine them closely and become involved in the on-going definition process.

All Version 1.0 Specifications are currently available to EPCglobal Subscribers and the general public. EPCglobal will launch a solution center for subscribers during the summer of 2004 to access board ratified standards and other implementation-related information information to enable the adoption of the EPCglobal Network. Look for more information coming soon.

NOTE: Click the below titles to download Version 1.0 Specification files.

**Version 1.0 Specifications**

1. **EPC Tag Data Specification Version 1.1** -- This EPCglobal Board Ratified specification identifies the specific encoding schemes for a serialized version of the EAN.UCC Global Trade Item Number (GTIN®), the EAN.UCC Serial Shipping Container Code (SSCC®), the EAN.UCC Global Location Number (GLN®), the EAN.UCC Global Returnable Asset Identifier (GRAI®), the EAN.UCC Global Individual Asset Identifier (GIAI®), and a General Identifier (GID).
2. **900 MHz Class 0 Radio Frequency (RF) Identification Tag Specification.** This document specifies the communications interface and protocol for 900 MHz Class 0 operation. It includes the RF and tag requirements and provides operational algorithms to enable communications in this band.
3. **13.56 MHz ISM Band Class 1 Radio Frequency (RF) Identification Tag Interface Specification.** This specification defines the communications interface and protocol for 13.56 MHz Class 1 operation. It also includes the RF and tag requirements to enable communications in this band.
4. **860MHz -- 930 MHz Class 1 Radio Frequency (RF) Identification Tag Radio Frequency & Logical Communication Interface Specification.** This document specifies the communications interface and protocol for 860 - 930 MHz Class 1 operation. It includes the RF and tag requirements to enable communications in this band.
5. **Reader Protocol**. This specification defines the communications messaging and protocol between tag readers and EPC compliant software applications, including Savant.
6. **Savant Specification.** This specification defines the services Savant performs for application requests within the EPCglobal Network.
7. **Physical Markup Language (PML) Core Specification, Extensible Markup Language (XML) Schema and Instance Files.** The PML Core specification establishes a common vocabulary set to be used within the EPCglobal Network. It provides a standardised format for data captured by readers. This specification also includes XML Schema and Instance files for your reference.
8. **Object Name Service (ONS) Specification.** This document specifies how the ONS is used to locate authoritative meta-data and services associated with a given Electronic Product Code (EPC).



© 2003 EPCglobal, Inc. - A joint venture between EAN International and the Uniform Code Council, Inc.

# Exhibit EPC02

DONATE

INTERNET ARCHIVE WayBack Machine

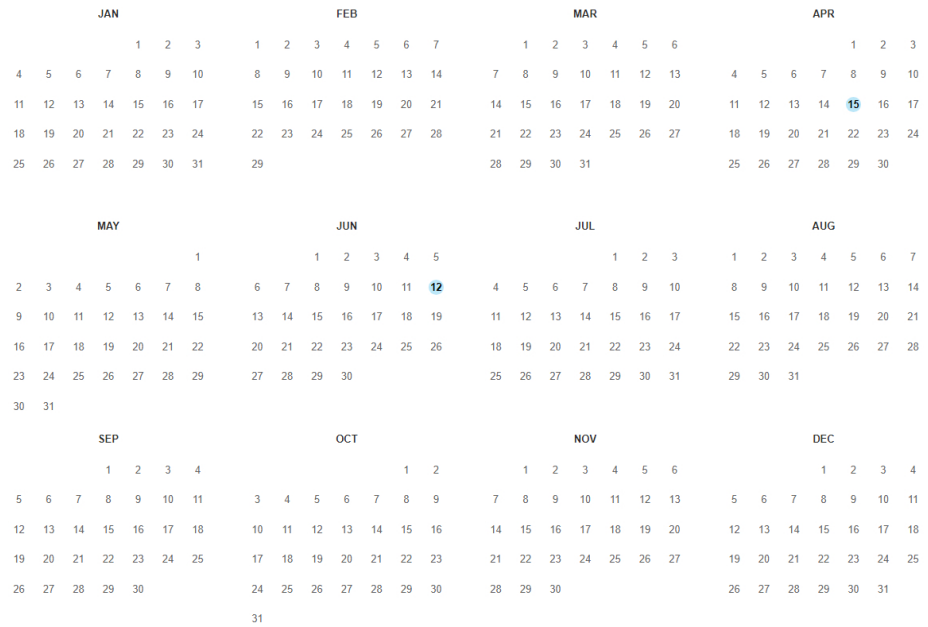
Explore more than 866 billion web pages saved over time

http://www.epcglobalinc.org:80/standards_technology/EPCTagDataSpe

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **11 times** between April 15, 2004 and September 4, 2014.

Sat, 12 Jun 2004 22:22:24 GMT (why: alexa_dx, alexacrawls)




## Note

This calendar view maps the number of times **http://www.epcglobalinc.org:80/standards_technology/EPCTagDataSpecification11rev124.pdf** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.



The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

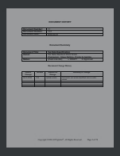
Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use.

https://web.archive.org/web/20040612222224/http://www.epcglobalinc.org:80/standards_technology/EPCTagDataSpecification11rev124.pdf 2024-08-26T13:38:01-04:00

INTERNET ARCHIVE WayBackMachine
http://www.epcglobalinc.org/standards_technology/EPCTagDataSpecification11rev124.pdf Go
11 captures
15 Apr 2004 - 4 Sep 2014
APR JUN 12 2004 APR
2003 2005
About this capture

EPC Tag Data Specification 1.1 Rev 1.241.doc 1 / 78 90%

1


2

3

EPCglobal

# EPC™ Tag Data Standards Version 1.1 Rev.1.24

Standard Specification
01 April 2004

# EPC™ Tag Data Standards Version 1.1 Rev.1.24

Standard Specification
01 April 2004

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## DOCUMENT HISTORY

| | |
|---|---|
| **Document Number:** | 1.1 |
| **Document Version:** | 1.24 |
| **Document Date :** | 2004-04-01 |

## Document Summary

| | |
|---|---|
| **Document Title:** | **Tag Data Specification** |
| **Owner:** | Tag Data Standard Work Group<br>Chairperson: Steve Rehling, Procter & Gamble |
| **Status:** | (*check one box*) ✍ DRAFT X Approved |

## Document Change History

| Date of Change | Version | Reason for Change | Summary of Change |
|---|---|---|---|
| 03-31-2004 | 1.24 | Update errata | Comments and errata identified during public review |
| | | | |
| | | | |
| | | | |

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# Abstract

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

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

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

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

# Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the EPCglobal. This document is the Last Call Working Draft.

This is an EPCglobal Center Working Draft for review by EPCglobal Members and other interested parties. It is a draft document and may be updated, replaced or made obsolete by other documents at any time. It is inappropriate to use EPCglobal Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by the EPCglobal membership.

Comments on this document should be sent to the EPCglobal Software Action Group mailing list sag-tds@develop.autoidcenter.org.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## Changes from Previous Versions

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# Table of Contents


1 Introduction ..... 8
2 Identity Concepts ..... 9
2.1 Pure Identities ..... 10
2.1.1 General Types ..... 10
2.1.2 EAN.UCC System Identity Types ..... 11
2.1.2.1 Serialized Global Trade Identification Number (SGTIN) ..... 12
2.1.2.2 Serial Shipping Container Code (SSCC) ..... 13
2.1.2.3 Serialized Global Location Number (SGLN) ..... 13
2.1.2.4 Global Returnable Asset Identifier (GRAI) ..... 14
2.1.2.5 Global Individual Asset Identifier (GIAI) ..... 15
3 EPC Tag Bit-level Encodings ..... 15
3.1 Headers ..... 16
3.2 Notational Conventions ..... 18
3.3 General Identifier (GID-96) ..... 19
3.3.1.1 GID-96 Encoding Procedure ..... 19
3.3.1.2 GID-96 Decoding Procedure ..... 20
3.4 Serialized Global Trade Item Number (SGTIN) ..... 20
3.4.1 SGTIN-64 ..... 20
3.4.1.1 SGTIN-64 Encoding Procedure ..... 22
3.4.1.2 SGTIN-64 Decoding Procedure ..... 22
3.4.2 SGTIN-96 ..... 23
3.4.2.1 SGTIN-96 Encoding Procedure ..... 24
3.4.2.2 SGTIN-96 Decoding Procedure ..... 25
3.5 Serial Shipping Container Code (SSCC) ..... 26
3.5.1 SSCC-64 ..... 26
3.5.1.1 SSCC-64 Encoding Procedure ..... 27
3.5.1.2 SSCC-64 Decoding Procedure ..... 28
3.5.2 SSCC-96 ..... 28


*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

3.5.2.1 SSCC-96 Encoding Procedure ..... 30
3.5.2.2 SSCC-96 Decoding Procedure ..... 30
3.6 Serialized Global Location Number (SGLN) ..... 31
3.6.1 SGLN-64 ..... 32
3.6.1.1 SGLN-64 Encoding Procedure ..... 33
3.6.1.2 SGLN-64 Decoding Procedure ..... 33
3.6.2 SGLN-96 ..... 34
3.6.2.1 SGLN-96 Encoding Procedure ..... 35
3.6.2.2 SGLN-96 Decoding Procedure ..... 36
3.7 Global Returnable Asset Identifier (GRAI) ..... 37
3.7.1 GRAI-64 ..... 37
3.7.1.1 GRAI-64 Encoding Procedure ..... 38
3.7.1.2 GRAI-64 Decoding Procedure ..... 39
3.7.2 GRAI-96 ..... 40
3.7.2.1 GRAI-96 Encoding Procedure ..... 41
3.7.2.2 GRAI-96 Decoding Procedure ..... 42
3.8 Global Individual Asset Identifier (GIAI) ..... 43
3.8.1 GIAI-64 ..... 43
3.8.1.1 GIAI-64 Encoding Procedure ..... 44
3.8.1.2 GIAI-64 Decoding Procedure ..... 45
3.8.2 GIAI-96 ..... 45
3.8.2.1 GIAI-96 Encoding Procedure ..... 47
3.8.2.2 GIAI-96 Decoding Procedure ..... 47
4 URI Representation ..... 48
4.1 URI Forms for Pure Identities ..... 49
4.2 URI Forms for Related Data Types ..... 50
4.2.1 URIs for EPC Tags ..... 51
4.2.2 URIs for Raw Bit Strings Arising From Invalid Tags ..... 51
4.2.3 URIs for EPC Patterns ..... 52
4.3 Syntax ..... 53
4.3.1 Common Grammar Elements ..... 53
4.3.2 EPCGID-URI ..... 53

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

4.3.3 SGTIN-URI ..... 53
4.3.4 SSCC-URI ..... 53
4.3.5 SGLN-URI ..... 54
4.3.6 GRAI-URI ..... 54
4.3.7 GIAI-URI ..... 54
4.3.8 EPC Tag URI ..... 54
4.3.9 Raw Tag URI ..... 55
4.3.10 EPC Pattern URI ..... 55
4.3.11 Summary (non-normative) ..... 55
5 Translation between EPC-URI and Other EPC Representations ..... 57
6 Semantics of EPC Pattern URIs ..... 64
7 Background Information ..... 64
8 References ..... 66
9 Appendix A: Encoding Scheme Summary Tables ..... 67
10 Appendix B: EPC Header Values and Tag Identity Lengths ..... 72
11 Appendix C: Example of a Specific Trade Item (SGTIN) ..... 74
12 Appendix D: Binary Digit Capacity Tables ..... 76
13 Appendix E: List of Abbreviations ..... 77
14 Appendix F: General EAN.UCC Specifications ..... 78
15 General EAN.UCC Specifications ..... **Error! Bookmark not defined.**
15.1 Section 3.0 Definition of Element Strings ..... 78
15.2 Section 3.7 EPCglobal Tag Data Standard ..... 78

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# 1 Introduction

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

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

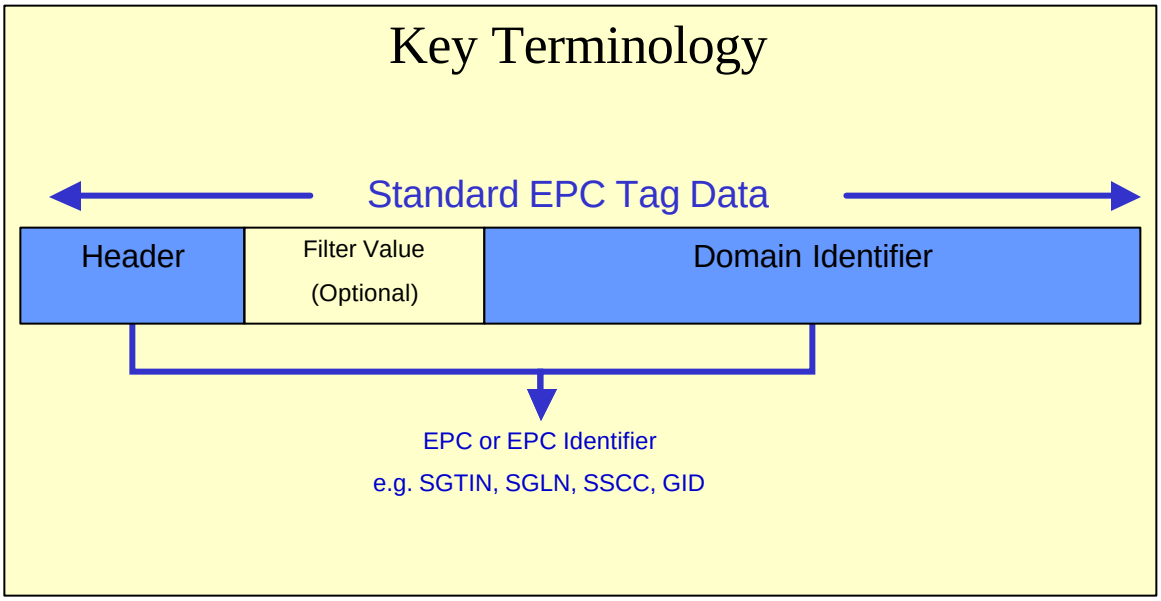



**Figure A.** EPC Terminology.

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

EAN.UCC Global Location Number (GLN®), the EAN.UCC Global Returnable Asset Identifier (GRAI®), and the EAN.UCC Global Individual Asset Identifier (GIAI®).

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

The EPCglobal Tag Data Standard V1.1 R1.23 has been approved by EAN.UCC with the restrictions outlined in the General EAN.UCC Specifications Section 3.7, which is excerpted into Tag Data Standard Appendix F.

# 2 Identity Concepts

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

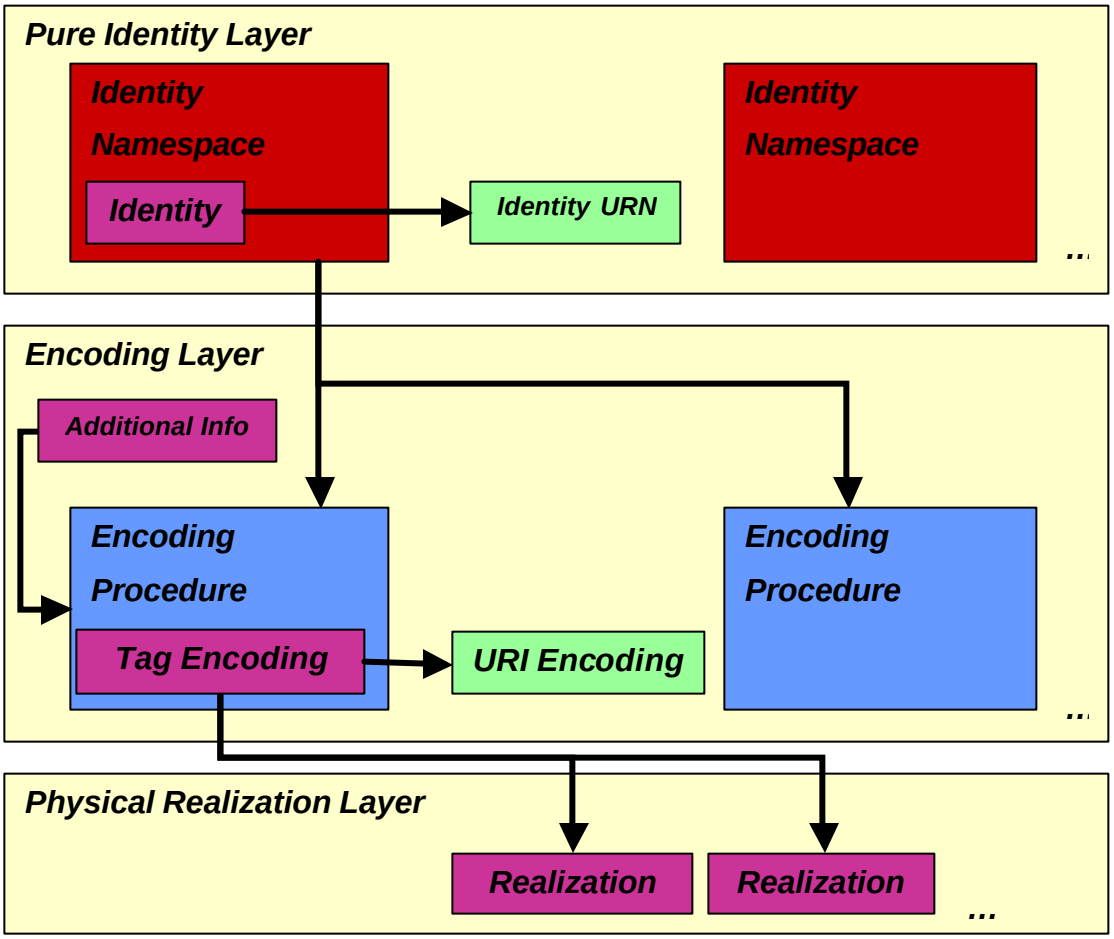



**Figure B.** Defined Identity Namespaces, Encodings, and Realizations.

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. 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.

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.

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

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

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

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

## 2.1 Pure Identities

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

### 2.1.1 General Types

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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

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

### 2.1.2 EAN.UCC System Identity Types

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

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

#### 2.1.2.1 Serialized Global Trade Identification Number (SGTIN)

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

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

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

The SGTIN consists of the following information elements:

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

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

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

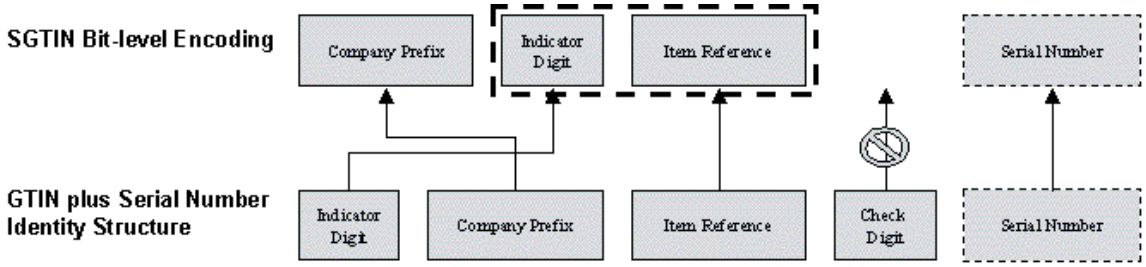



**Figure C.** How the parts of the decimal SGTIN are extracted, rearranged, and augmented for encoding.

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

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

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

The SSCC consists of the following information elements:

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

The *Serial Reference*, assigned uniquely by the managing entity to a specific shipping unit. The Serial Reference for the purposes of EPC encoding is derived from the SSCC by concatenating the Extension Digit of the SSCC and the Serial Reference digits, and treating the result as a single integer.

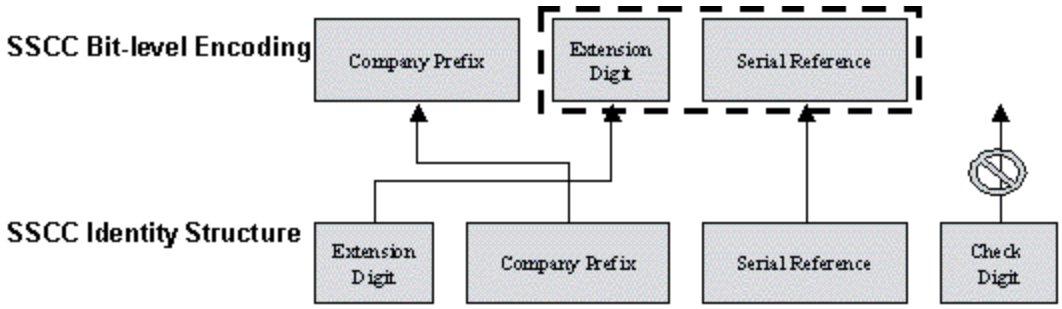



**Figure D.** How the parts of the decimal SSCC are extracted and rearranged for encoding.

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

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

The SGLN consists of the following information elements:

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

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

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

✍✍The serial number should not be used until specified by the EAN.UCC General Specifications .

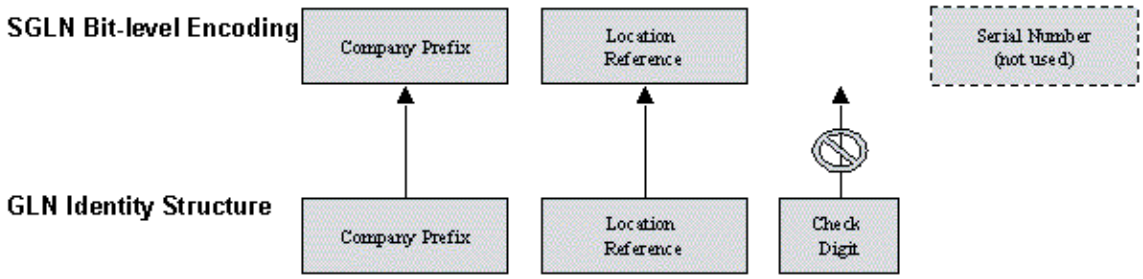



**Figure E.** How the parts of the decimal SGLN are extracted and rearranged for encoding.

#### 2.1.2.4 Global Returnable Asset Identifier (GRAI)

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

The GRAI consists of the following information elements:

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

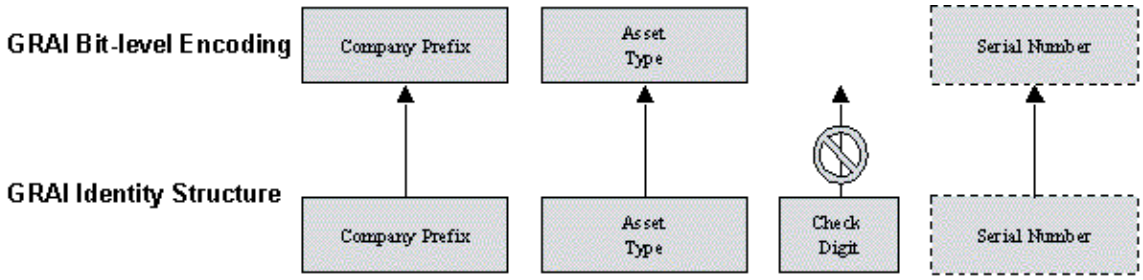



**Figure F.** How the parts of the decimal GRAI are extracted and rearranged for encoding.

#### 2.1.2.5 Global Individual Asset Identifier (GIAI)

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

The GIAI consists of the following information elements:

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

The *Individual Asset Reference*, assigned uniquely by the managing entity to a specific asset. The EPC representation is only capable of representing a subset of Individual Asset References allowed in the General EAN.UCC Specifications. Specifically, only those Individual Asset References consisting of one or more digits, with no leading zeros, are permitted.

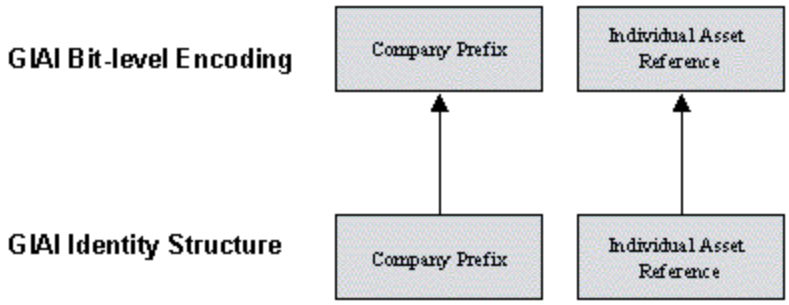



**Figure G.** How the parts of the decimal GIAI are extracted and rearranged for encoding.

# 3 EPC Tag Bit-level Encodings

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.*

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

## 3.1 Headers

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

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

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

| **Header Value (binary)** | **Tag Length (bits)** | **EPC Encoding Scheme** |
|---|---|---|
| 01 | 64 | [Reserved 64-bit scheme] |
| 10 | 64 | SGTIN-64 |
| 11 | 64 | [Reserved 64-bit scheme] |
| *0000 0001*<br>*0000 001x*<br>*0000 01xx* | *na*<br>*na*<br>*na* | *[1 reserved scheme]*<br>*[2 reserved schemes]*<br>*[4 reserved schemes]* |
| 0000 1000 | 64 | SSCC-64 |
| 0000 1001 | 64 | GLN-64 |
| 0000 1010 | 64 | GRAI-64 |
| 0000 1011 | 64 | GIAI-64 |
| 0000 1100<br>…<br>0000 1111 | 64 | [4 reserved 64-bit schemes] |
| 0001 0000<br>…<br>0010 1111 | na | [32 reserved schemes] |
| 0011 0000 | 96 | SGTIN-96 |
| 0011 0001 | 96 | SSCC-96 |
| 0011 0010 | 96 | GLN-96 |
| 0011 0011 | 96 | GRAI-96 |
| 0011 0100 | 96 | GIAI-96 |
| 0011 0101 | 96 | GID-96 |
| 0011 0110<br>…<br>0011 1111 | 96 | [10 reserved 96-bit schemes] |
| 0000 0000 … | | [reserved for future headers longer than 8 bits] |

**Table 1.** Electronic Product Code Headers

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## 3.2 Notational Conventions

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

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

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

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

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

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

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## 3.3 General Identifier (GID-96)

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

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

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

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

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

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

#### 3.3.1.1 GID-96 Encoding Procedure

The following procedure creates a GID-96 encoding.

Given:

An General Manager Number $M$ where $0 = M < 2^{28}$

An Object Class $C$ where $0 = C < 2^{24}$

A Serial Number $S$ where $0 = S < 2^{36}$

Procedure:

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

#### 3.3.1.2 GID-96 Decoding Procedure

Given:

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

Yields:

An General Manager Number

An Object Class

A Serial Number

Procedure:

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

## 3.4 Serialized Global Trade Item Number (SGTIN)

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

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

### 3.4.1 SGTIN-64

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

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

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

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

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

*Filter Value* is not part of the SGTIN pure identity, but is additional data that is used for fast filtering and pre-selection of basic logistics types, such as items, inner packs, cases and pallets. The Filter Values for 64-bit and 96-bit SGTIN are the same. The normative specifications for Filter Values are undefined at this time; see Table 5 for a non-normative summary (for purposes of illustration only).

| Type | Binary Value |
|---|---|
| Other | xxx |
| Item | xxx |
| Inner Pack | xxx |
| Case | xxx |
| Load/Pallet | xxx |
| Reserved | xxx |

**Table 5.** SGTIN Filter Values (non-normative).

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

*Item Reference* encodes the GTIN Item Reference number and Indicator Digit. 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*

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

*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.

*Serial Number* contains a serial number. The 25-bit capacity is limited to serial numbers up to 33,554,431, smaller than the EAN.UCC System specification for serial number. Only numbers are permitted.

#### 3.4.1.1 SGTIN-64 Encoding Procedure

The following procedure creates an SGTIN-64 encoding.

Given:

An EAN.UCC GTIN-14 consisting of digits $d_1 d_2 \ldots d_{14}$

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

A Serial Number $S$ where $0 = S < 2^{25}$

A Filter Value $F$ where $0 = F < 8$

Procedure:

1. Extract the EAN.UCC Company Prefix $d_2 d_3 \ldots d_{(L+1)}$

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

3. Construct the Item Reference by concatenating digits $d_1 d_{(L+2)} d_{(L+3)} \ldots d_{13}$ and considering the result to be a decimal integer, $I$. If $I = 2^{20}$, stop: this GTIN cannot be encoded in the SGTIN-64 encoding.

4. Construct the final encoding by concatenating the following bit fields, from most significant to least significant: Header 10 (2 bits), Filter Value $F$ (3 bits), Company Prefix Index $C$ from Step 2 (14 bits), Item Reference from Step 3 (20 bits), Serial Number $S$ (25 bits).

#### 3.4.1.2 SGTIN-64 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GTIN-14

A Serial Number

A Filter Value

Procedure:

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

3. Look up the Company Prefix Index $C$ in the Company Prefix Translation Table to obtain the EAN.UCC Company Prefix $p_1p_2\ldots p_L$ consisting of L decimal digits (the value of L is also obtained from the table).

4. Consider bits $b_{44}b_{43}\ldots b_{25}$ as an unsigned integer. If this integer is greater than or equal to $10^{(13-L)}$, stop: the input bit string is not a legal SGTIN-64 encoding. Otherwise, convert this integer to a (13-L)-digit decimal number $i_1i_2\ldots i_{(13-L)}$, adding leading zeros as necessary to make (13-L) digits.

5. Construct a 13-digit number $d_1d_2\ldots d_{13}$ where $d_1 = i_1$ from Step 4, $d_2d_3\ldots d_{(L+1)} = p_1p_2\ldots p_L$ from Step 3, and $d_{(L+2)}d_{(L+3)}\ldots d_{13} = i_2\ i_3\ldots i_{(13-L)}$ from Step 4.

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

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

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

### 3.4.2 SGTIN-96

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

| | Header | Filter Value | Partition | Company Prefix | Item Reference | Serial Number |
|---|---|---|---|---|---|---|
| SGTIN-96 | 8 | 3 | 3 | 20-40 | 24-4 | 38 |
| | 0011 0000<br>(Binary value) | 8<br>(Decimal capacity) | 8<br>(Decimal capacity) | 999,999 – 999,999,999,999<br>(Decimal capacity*) | 9,999,999 – 9<br>(Decimal capacity*) | 274,877,906,943<br>(Decimal capacity) |

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

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

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

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

*Partition* is an indication of where the subsequent Company Prefix and Item Reference numbers are divided. This organization matches the structure in the EAN.UCC GTIN in which the Company Prefix added to the Item Reference number (plus the single Indicator Digit) totals 13 digits, yet the Company Prefix may vary from 6 to 12 digits and the Item

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

Reference (including the single Indicator Digit) 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 7.

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

*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.

*Serial Number* contains a serial number. The capacity of this serial number is less than the maximum EAN.UCC System specification for serial number, and only numbers are permitted.

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

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

#### 3.4.2.1 SGTIN-96 Encoding Procedure

The following procedure creates an SGTIN-96 encoding.

Given:

An EAN.UCC GTIN-14 consisting of digits $d_1 d_2 \ldots d_{14}$

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

A Serial Number $S$ where $0 = S < 2^{38}$

A Filter Value $F$ where $0 = F < 8$

Procedure:

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

2. Construct the Company Prefix by concatenating digits $d_2 d_3 \ldots d_{(L+1)}$ and considering the result to be a decimal integer, $C$.

3. Construct the Item Reference by concatenating digits $d_1 d_{(L+2)} d_{(L+3)} \ldots d_{13}$ and considering the result to be a decimal integer, $I$.

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

#### 3.4.2.2 SGTIN-96 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GTIN-14

A Serial Number

A Filter Value

Procedure:

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

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

3. Look up the Partition Value $P$ in Table 7 to obtain the number of bits $M$ in the Company Prefix and the number of digits $L$ in the Company Prefix.

4. Extract the Company Prefix $C$ by considering bits $b_{81} b_{80} \ldots b_{(82-M)}$ as an unsigned integer. If this integer is greater than or equal to $10^L$, stop: the input bit string is not a legal SGTIN-96 encoding. Otherwise, convert this integer into a decimal number $p_1 p_2 \ldots p_L$, adding leading zeros as necessary to make up $L$ digits in total.

5. Extract the Item Reference and Indicator by considering bits $b_{(81-M)}\, b_{(80-M)} \ldots b_{38}$ as an unsigned integer. If this integer is greater than or equal to $10^{(13-L)}$, stop: the input bit string is not a legal SGTIN-96 encoding. Otherwise, convert this integer to a (13-L)-digit decimal number $i_1 i_2 \ldots i_{(13-L)}$, adding leading zeros as necessary to make (13-L) digits.

6. Construct a 13-digit number $d_1 d_2 \ldots d_{13}$ where $d_1 = i_1$ from Step 5, $d_2 d_3 \ldots d_{(L+1)} = p_1 p_2 \ldots p_L$ from Step 4, and $d_{(L+2)} d_{(L+3)} \ldots d_{13} = i_2\, i_3 \ldots i_{(13-L)}$ from Step 5.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

8. The EAN.UCC GTIN-14 is the concatenation of digits from Steps 6 and 7: $d_1 d_2 \ldots d_{14}$.

9. Bits $b_{37} b_{36} \ldots b_0$, considered as an unsigned integer, are the Serial Number.

## 3.5 Serial Shipping Container Code (SSCC)

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

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

### 3.5.1 SSCC-64

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

| | Header | Filter Value | Company Prefix *Index* | Serial Reference |
|---|---|---|---|---|
| SSCC-64 | 8 | 3 | 14 | 39 |
| | 0000 1000 (Binary value) | 8 (Decimal capacity) | 16,383 (Decimal capacity) | 99,999 - 99,999,999,999 (Decimal capacity*) |

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

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

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

*Filter Value* is not part of the SSCC or EPC identifier, but is used for fast filtering and pre-selection of basic logistics types, such as cases and pallets. The Filter Values for 64-bit and 96-bit SSCC are the same. The normative specifications for Filter Values are undefined at this time; see Table 9 for a non-normative summary (for purposes of illustration only).

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

| Type | Binary Value |
|---|---|
| Other | xxx |
| Case | xxx |
| Load/Pallet | xxx |
| Reserved | xxx |

**Table 9.** SSCC Filter Values (non-normative).

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

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

#### 3.5.1.1 SSCC-64 Encoding Procedure

The following procedure creates an SSCC-64 encoding.

Given:

An EAN.UCC SSCC consisting of digits $d_1 d_2 \ldots d_{18}$

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

A Filter Value $F$ where $0 = F < 8$

Procedure:

1. Extract the EAN.UCC Company Prefix $d_2 d_3 \ldots d_{(L+1)}$

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

3. Construct the Serial Reference by concatenating digits $d_1 d_{(L+2)} d_{(L+3)} \ldots d_{17}$ and considering the result to be a decimal integer, $I$. If $I = 2^{39}$, stop: this SSCC cannot be encoded in the SSCC-64 encoding.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

4. Construct the final encoding by concatenating the following bit fields, from most significant to least significant: Header 00001000 (8 bits), Filter Value $F$ (3 bits), Company Prefix Index $C$ from Step 2 (14 bits), Serial Reference from Step 3 (39 bits).

#### 3.5.1.2 SSCC-64 Decoding Procedure

Given:

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

Yields:

An EAN.UCC SSCC

A Filter Value

Procedure:

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

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

3. Look up the Company Prefix Index $C$ in the Company Prefix Translation Table to obtain the EAN.UCC Company Prefix $p_1p_2\ldots p_L$ consisting of L decimal digits (the value of L is also obtained from the table).

4. Consider bits $b_{38}b_{37}\ldots b_0$ as an unsigned integer. If this integer is greater than or equal to $10^{(17-L)}$, stop: the input bit string is not a legal SSCC-64 encoding. Otherwise, convert this integer to a (17-L)-digit decimal number $i_1i_2\ldots i_{(17-L)}$, adding leading zeros as necessary to make (17-L) digits.

5. Construct a 17-digit number $d_1d_2\ldots d_{17}$ where $d_1 = s_1$ from Step 4, $d_2d_3\ldots d_{(L+1)} = p_1p_2\ldots p_L$ from Step 3, and $d_{(L+2)}d_{(L+3)}\ldots d_{17} = i_2\, i_3\ldots i_{(17-L)}$ from Step 4.

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

7. The EAN.UCC SSCC is the concatenation of digits from Steps 5 and 6: $d_1d_2\ldots d_{18}$.

### 3.5.2 SSCC-96

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

| | Header | Filter Value | Partition | Company Prefix | Serial Reference | Unallocated |
|---|---|---|---|---|---|---|
| SSCC-96 | 8 | 3 | 3 | 20-40 | 37-17 | 25 |
| | 0011 0001<br>(Binary value) | 8<br>(Decimal capacity) | 8<br>(Decimal capacity) | 999,999 – 999,999,999,999<br>(Decimal capacity*) | 99,999,999,999 – 99,999<br>(Decimal capacity*) | [Not Used] |

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

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

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

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

The *Partition* is an indication of where the subsequent Company Prefix and Serial Reference numbers are divided. This organization matches the structure in the EAN.UCC SSCC in which the Company Prefix added to the Serial Reference number (including 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 digit(s). Table 11 shows allowed values of the partition value and the corresponding lengths of the company prefix and serial reference.

| **Partition Value (*P*)** | **Company Prefix** | | **Serial Reference and Extension Digit** | |
|---|---|---|---|---|
| | **Bits (*M*)** | **Digits (*L*)** | **Bits (*N*)** | **Digits** |
| 0 | 40 | 12 | 17 | 5 |
| 1 | 37 | 11 | 20 | 6 |
| 2 | 34 | 10 | 24 | 7 |
| 3 | 30 | 9 | 27 | 8 |
| 4 | 27 | 8 | 30 | 9 |
| 5 | 24 | 7 | 34 | 10 |
| 6 | 20 | 6 | 37 | 11 |

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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

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

#### 3.5.2.1 SSCC-96 Encoding Procedure

The following procedure creates an SSCC-96 encoding.

Given:

An EAN.UCC SSCC consisting of digits $d_1 d_2 \ldots d_{18}$

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

A Filter Value $F$ where $0 = F < 8$

Procedure:

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

2. Construct the Company Prefix by concatenating digits $d_2 d_3 \ldots d_{(L+1)}$ and considering the result to be a decimal integer, $C$.

3. Construct the Serial Reference by concatenating digits $d_1 d_{(L+2)} d_{(L+3)} \ldots d_{17}$ and considering the result to be a decimal integer, $S$.

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 25 zero bits. Note that $M+N = 57$ bits for all $P$.

#### 3.5.2.2 SSCC-96 Decoding Procedure

Given:

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

Yields:

An EAN.UCC SSCC

A Filter Value

Procedure:

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

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

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

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

5. Extract the Serial Reference by considering bits $b_{(81-M)}\, b_{(80-M)}\ldots b_{25}$ as an unsigned integer. If this integer is greater than or equal to $10^{(17-L)}$, stop: the input bit string is not a legal SSCC-96 encoding. Otherwise, convert this integer to a (17-L)-digit decimal number $i_1i_2\ldots i_{(17-L)}$, adding leading zeros as necessary to make (17-L) digits.

6. Construct a 17-digit number $d_1d_2\ldots d_{17}$ where $d_1 = s_1$ from Step 5, $d_2d_3\ldots d_{(L+1)} = p_1p_2\ldots p_L$ from Step 4, and $d_{(L+2)}d_{(L+3)}\ldots d_{17} = i_2\, i_3\ldots i_{(17-L)}$ from Step 5.

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

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

## 3.6 Serialized Global Location Number (SGLN)

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

### 3.6.1 SGLN-64

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

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

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

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

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

*Filter Value* is not part of the SGLN pure identity, but is additional data that is used for fast filtering and pre-selection of basic location types. The Filter Values for 64-bit and 96-bit SGLN are the same. The normative specifications for Filter Values are undefined at this time; see Table 13 for a non-normative summary (for purposes of illustration only).

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

*Location Reference* encodes the GLN Location Reference number.

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

| Type | Binary Value |
|---|---|
| Other | xxx |
| Physical Location | xxx |
| Reserved | xxx |

**Table 13.** SGLN Filter Values (non-normative).

#### 3.6.1.1 SGLN-64 Encoding Procedure

The following procedure creates an SGLN-64 encoding.

Given:

An EAN.UCC GLN consisting of digits $d_1 d_2 \ldots d_{13}$

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

A Serial Number $S$ where $0 = S < 2^{19}$

A Filter Value $F$ where $0 = F < 8$

Procedure:

1. Extract the EAN.UCC Company Prefix $d_1 d_2 \ldots d_L$

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

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

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

#### 3.6.1.2 SGLN-64 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GLN

A Serial Number

A Filter Value

Procedure:

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

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

3. Look up the Company Prefix Index $C$ in the Company Prefix Translation Table to obtain the EAN.UCC Company Prefix $p_1p_2\ldots p_L$ consisting of L decimal digits (the value of L is also obtained from the table).

4. Consider bits $b_{38}b_{37}\ldots b_{19}$ as an unsigned integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a legal SGLN-64 encoding. Otherwise, convert this integer to a (12-L)-digit decimal number $i_1i_2\ldots i_{(12-L)}$, adding leading zeros as necessary to make (12-L) digits.

5. Construct a 12-digit number $d_1d_2\ldots d_{12}$ where $d_1d_2\ldots d_L = p_1p_2\ldots p_L$ from Step 3, and $d_{(L+1)}d_{(L+2)}\ldots d_{12} = i_1\ i_2\ldots i_{(12-L)}$ from Step 4.

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

7. The EAN.UCC GLN is the concatenation of digits from Steps 5 and 6: $d_1d_2\ldots d_{13}$.

8. Bits $b_{18}b_{17}\ldots b_0$, considered as an unsigned integer, are the Serial Number.

### 3.6.2 SGLN-96

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

*Header* is 8-bits, with a binary value of 0011 0010.

*Filter Value* is not part of the GLN or EPC identifier, but is used for fast filtering and pre-selection of basic location types. The Filter Values for 64-bit and 96-bit GLN are the same. See Table 13.

*Partition* is an indication of where the subsequent Company Prefix and Location Reference numbers are divided. This organization matches the structure in the EAN.UCC GLN in which the Company Prefix added to the Location Reference number totals 12 digits, yet the Company Prefix may vary from 6 to 12 digits and the Location Reference number from 6 to 0 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Location Reference* fields are defined in Table 15.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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

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

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

*Location Reference* encodes the GLN Location Reference number.

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

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

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

#### 3.6.2.1 SGLN-96 Encoding Procedure

The following procedure creates an SGLN-96 encoding.

Given:

An EAN.UCC GLN consisting of digits $d_1 d_2 \ldots d_{13}$

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

A Serial Number $S$ where $0 = S < 2^{41}$

A Filter Value $F$ where $0 = F < 8$

Procedure:

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

2. Construct the Company Prefix by concatenating digits $d_1 d_2 \ldots d_L$ and considering the result to be a decimal integer, $C$.

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

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

#### 3.6.2.2 SGLN-96 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GLN

A Serial Number

A Filter Value

Procedure:

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

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

3. Look up the Partition Value $P$ in Table 15 to obtain the number of bits $M$ in the Company Prefix and the number of digits $L$ in the Company Prefix.

4. Extract the Company Prefix $C$ by considering bits $b_{81} b_{80} \ldots b_{(82-M)}$ as an unsigned integer. If this integer is greater than or equal to $10^L$, stop: the input bit string is not a legal SGLN-96 encoding. Otherwise, convert this integer into a decimal number $p_1 p_2 \ldots p_L$, adding leading zeros as necessary to make up $L$ digits in total.

5. Extract the Location Reference by considering bits $b_{(81-M)}\, b_{(80-M)} \ldots b_{41}$ as an unsigned integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a legal SGLN-96 encoding. Otherwise, convert this integer to a (12- L)-digit decimal number $i_1 i_2 \ldots i_{(12-L)}$, adding leading zeros as necessary to make (12- L) digits.

6. Construct a 12-digit number $d_1 d_2 \ldots d_{12}$ where $d_1 d_2 \ldots d_L = p_1 p_2 \ldots p_L$ from Step 4, and $d_{(L+1)} d_{(L+2)} \ldots d_{12} = i_2\, i_3 \ldots i_{(12-L)}$ from Step 5.

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

8. The EAN.UCC GLN is the concatenation of digits from Steps 6 and 7: $d_1 d_2 \ldots d_{13}$.

9. Bits $b_{40} b_{39} \ldots b_0$, considered as an unsigned integer, are the Serial Number.

## 3.7 Global Returnable Asset Identifier (GRAI)

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

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

### 3.7.1 GRAI-64

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

| | Header | Filter Value | Company Prefix Index | Asset Type | Serial Number |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| | | | Index | | |
| GRAI-64 | 8 | 3 | 14 | 20 | 19 |
| | 0000 1010 (Binary value) | 8 (Decimal capacity) | 16,383 (Decimal capacity) | 9,999,999 - 9 (Decimal capacity*) | 524,288 (Decimal capacity) |

*Capacity of Asset Type field varies with Company Prefix.

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

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

*Filter Value* is not part of the GRAI pure identity, but is additional data that is used for fast filtering and pre-selection of basic asset types. The Filter Values for 64-bit and 96-bit GRAI are the same. The Filter Values for GRAI are undefined at this time. However, this specification anticipates that valuable Filter Values will be determined once there has been time to consider the possible use cases.

| Type | Binary Value |
|---|---|
| tbd | tbd |
| Reserved | xxx |

**Table 17.** GRAI Filter Values (non-normative).

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

*Asset Type* encodes the GRAI Asset Type number.

*Serial Number* contains a serial number. The EPC representation is only capable of representing a subset of Serial Numbers allowed in the General EAN.UCC Specifications. The capacity of this mandatory serial number is less than the maximum EAN.UCC System specification for serial number, no leading zeros are permitted, and only numbers are permitted.

#### 3.7.1.1 GRAI-64 Encoding Procedure

The following procedure creates a GRAI-64 encoding.

Given:

An EAN.UCC GRAI consisting of digits $0d_2 \ldots d_K$, where 15 =K =30.

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

A Filter Value $F$ where $0 = F < 8$

Procedure:

1. Extract the EAN.UCC Company Prefix $d_2 d_3 \ldots d_{L+1}$

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

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

4. Construct the Serial Number by concatenating digits $d_{15} d_{16} \ldots d_K$ and considering the result to be a decimal integer, $S$. If $S = 2^{19}$, stop: this GRAI cannot be encoded in the GRAI-64 encoding. Also, if K > 15 and $d_{15} = 0$, stop: this GRAI cannot be encoded in the GRAI-64 encoding (because leading zeros are not permitted except in the case where the Serial Number consists of a single zero digit).

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

#### 3.7.1.2 GRAI-64 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GRAI

A Filter Value

Procedure:

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

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

3. Look up the Company Prefix Index $C$ in the Company Prefix Translation Table to obtain the EAN.UCC Company Prefix $p_1 p_2 \ldots p_L$ consisting of L decimal digits (the value of L is also obtained from the table).

4. Consider bits $b_{38} b_{37} \ldots b_{19}$ as an unsigned integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a legal GRAI-64 encoding. Otherwise,

convert this integer to a (12-L)-digit decimal number $i_1 i_2 \ldots i_{(12-L)}$, adding leading zeros as necessary to make (12-L) digits.

5. Construct a 13-digit number $0d_2d_3\ldots d_{13}$ where $d_2d_3\ldots d_{L+1} = p_1p_2\ldots p_L$ from Step 3, and $d_{(L+2)}d_{(L+3)}\ldots d_{13} = i_1\ i_2\ldots i_{(12-L)}$ from Step 4.

6. Calculate the check digit $d_{14} = (-3(d_3 + d_5 + d_7 + d_9 + d_{11}+d_{13}) - (d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12}))$ mod 10.

7. Consider bits $b_{18}b_{17}\ldots b_0$ as an unsigned integer. Convert this integer into a decimal number $d_{15}d_{16}\ldots d_K$, with no leading zeros (exception: if the integer is equal to zero, convert it to a single zero digit).

8. The EAN.UCC GRAI is the concatenation of the digits from Steps 5, 6, and 7: $0d_2d_3\ldots d_K$.

### 3.7.2 GRAI-96

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

| | Header | Filter Value | Partition | Company Prefix | Asset Type | Serial Number |
|---|---|---|---|---|---|---|
| GRAI-96 | 8 | 3 | 3 | 20-40 | 24-4 | 38 |
| | 0011 0011 (Binary value) | 8 (Decimal capacity) | 8 (Decimal capacity) | 999,999 – 999,999,999,999 (Decimal capacity*) | 9,999,999 – 9 (Decimal capacity*) | 274,877,906,943 (Decimal capacity) |

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

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

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

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

*Partition* is an indication of where the subsequent Company Prefix and Asset Type numbers are divided. This organization matches the structure in the EAN.UCC GRAI in which the Company Prefix added to the Asset Type number totals 13 digits, yet the Company Prefix may vary from 6 to 12 digits and the Asset Type from 7 to 1 digit(s). The available values of *Partition* and the corresponding sizes of the *Company Prefix* and *Asset Type* fields are defined in Table 19.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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

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

*Asset Type* encodes the GRAI Asset Type number.

*Serial Number* contains a serial number. The EPC representation is only capable of representing a subset of Serial Numbers allowed in the General EAN.UCC Specifications. The capacity of this mandatory serial number is less than the maximum EAN.UCC System specification for serial number, no leading zeros are permitted, and only numbers are permitted.

#### 3.7.2.1 GRAI-96 Encoding Procedure

The following procedure creates a GRAI-96 encoding.

Given:

An EAN.UCC GRAI consisting of digits $0d_2d_3\ldots d_K$, where 15 =K =30.

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

A Filter Value $F$ where $0 = F < 8$

Procedure:

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

4. Construct the Serial Number by concatenating digits $d_{15}d_{16}\ldots d_K$ and considering the result to be a decimal integer, $S$. If $S = 2^{38}$, stop: this GRAI cannot be encoded in the GRAI-96 encoding. Also, if K > 15 and $d_{15} = 0$, stop: this GRAI cannot be encoded in the GRAI-96 encoding (because leading zeros are not permitted except in the case where the Serial Number consists of a single zero digit).

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

#### 3.7.2.2 GRAI-96 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GRAI

A Filter Value

Procedure:

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

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

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

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

5. Extract the Asset Type by considering bits $b_{(81-M)}\, b_{(80-M)}\ldots b_{38}$ as an unsigned integer. If this integer is greater than or equal to $10^{(12-L)}$, stop: the input bit string is not a legal GRAI-96 encoding. Otherwise, convert this integer to a (12-L)-digit decimal number $i_1i_2\ldots i_{(12-L)}$, adding leading zeros as necessary to make (12-L) digits.

6. Construct a 13-digit number $0d_2d_3\ldots d_{13}$ where $d_2d_3\ldots d_{(L+1)} = p_1p_2\ldots p_L$ from Step 4, and $d_{(L+2)}d_{(L+3)}\ldots d_{13} = i_1\, i_2\ldots i_{(12-L)}$ from Step 5.

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

8. Extract the Serial Number by considering bits $b_{37}b_{36}\ldots b_0$ as an unsigned integer. Convert this integer to a decimal number $d_{15}d_{16}\ldots d_K$, with no leading zeros (exception: if the integer is equal to zero, convert it to a single zero digit).

9. The EAN.UCC GRAI is the concatenation of a single zero digit and the digits from Steps 6, 7, and 8: $0d_2d_3\ldots d_K$.

## 3.8 Global Individual Asset Identifier (GIAI)

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

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

### 3.8.1 GIAI-64

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

| | Header | Filter Value | Company Prefix *Index* | Individual Asset Reference |
|---|---|---|---|---|
| GIAI-64 | 8 | 3 | 14 | 39 |
| | 0000 1011<br>(Binary value) | 8<br>(Decimal capacity) | 16,383<br>(Decimal capacity) | 549,755,813,888<br>(Decimal capacity) |

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

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

*Filter Value* is not part of the GIAI pure identity, but is additional data that is used for fast filtering and pre-selection of basic asset types. The Filter Values for 64-bit and 96-bit GIAI are the same. The Filter Values for GIAI are undefined at this time. However,

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

this specification anticipates that valuable Filter Values will be determined once there has been time to consider the possible use cases.

| Type | Binary Value |
|---|---|
| tbd | tbd |
| Reserved | xxx |

**Table 21.** GIAI Filter Values (non-normative).

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

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

#### 3.8.1.1 GIAI-64 Encoding Procedure

The following procedure creates a GIAI-64 encoding.

Given:

An EAN.UCC GIAI consisting of digits $d_1 d_2 \ldots d_K$ where K = 30.

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

A Filter Value $F$ where $0 = F < 8$

Procedure:

1. Extract the EAN.UCC Company Prefix $d_1 d_2 \ldots d_L$

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

3. Construct the Individual Asset Reference by concatenating digits $d_{(L+1)} d_{(L+2)} \ldots d_K$ and considering the result to be a decimal integer, $I$. If $I = 2^{39}$, stop: this GIAI cannot be encoded in the GIAI-64 encoding. Also, if K > L+1 and $d_{(L+1)} = 0$, stop: this GIAI cannot be encoded in the GIAI-64 encoding (because leading zeros are not permitted except in the case where the Individual Asset Reference consists of a single zero digit).

4. Construct the final encoding by concatenating the following bit fields, from most significant to least significant: Header 00001011 (8 bits), Filter Value $F$ (3 bits),

Company Prefix Index $C$ from Step 2 (14 bits), Individual Asset Reference from Step 3 (39 bits).

#### 3.8.1.2 GIAI-64 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GIAI

A Filter Value

Procedure:

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

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

3. Look up the Company Prefix Index $C$ in the Company Prefix Translation Table to obtain the EAN.UCC Company Prefix $p_1p_2\ldots p_L$ consisting of L decimal digits (the value of L is also obtained from the table).

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

5. Construct a K-digit number $d_1d_2\ldots d_K$ where $d_1d_2\ldots d_L = p_1p_2\ldots p_L$ from Step 3, and $d_{(L+1)}d_{(L+2)}\ldots d_K = s_1\, s_2\ldots s_J$ from Step 4. This K-digit number, where K = 30, is the EAN.UCC GIAI.

### 3.8.2 GIAI-96

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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

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

*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 64-bit and 96-bit GIAI are the same. See Table 21.

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 EAN.UCC 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 23.

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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

#### 3.8.2.1 GIAI-96 Encoding Procedure

The following procedure creates a GIAI-96 encoding.

Given:

An EAN.UCC GIAI consisting of digits $d_1 d_2 \ldots d_K$, where K = 30.

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

A Filter Value $F$ where $0 = F < 8$

Procedure:

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-96.

2. Construct the Company Prefix by concatenating digits $d_1 d_2 \ldots d_L$ and considering the result to be a decimal integer, $C$.

3. Construct the Individual Asset Reference by concatenating digits $d_{(L+1)} d_{(L+2)} \ldots d_K$ and considering the result to be a decimal integer, $S$. If $S = 2^N$, 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).

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

#### 3.8.2.2 GIAI-96 Decoding Procedure

Given:

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

Yields:

An EAN.UCC GIAI

A Filter Value

Procedure:

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

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

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

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

5. Extract the Individual Asset Reference by considering bits $b_{(81-M)}\, b_{(80-M)}\ldots b_0$ as an unsigned integer. If this integer is greater than or equal to $10^{(30-L)}$, stop: the input bit string is not a legal GIAI-96 encoding. Otherwise, convert this integer to a decimal number $s_1s_2\ldots s_J$, with no leading zeros (exception: if the integer is equal to zero, convert it to a single zero digit).

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

# 4 URI Representation

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

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## 4.1 URI Forms for Pure Identities

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

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

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

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

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

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

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

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

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

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

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

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

The `ItemReference`, `SerialReference`, `LocationReference`, and `AssetType` fields correspond to the similar fields of the GTIN, SSCC, GLN, and GRAI, respectively. Like the *`CompanyPrefix`* field, the number of digits in these fields is significant, and leading zeros are included as necessary. The number of digits in these fields, when added to the number of digits in the *`CompanyPrefix`* field, always total the same number of digits according to the identity type: 13 digits total for SGTIN, 17 digits total for SSCC, 12 digits total for SGLN, and 12 digits total for the GRAI. (The *`ItemReference`* field of the SGTIN includes the GTIN Indicator (PI) digit, appended to the beginning of the item reference. The *`SerialReference`* field includes the

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

SSCC Extension Digit (ED), appended at the beginning of the serial reference. In no case are check digits included in URI representations.)

In contrast to the other fields, the *`SerialNumber`* field of the SGTIN, SGLN, and GRAI, as well as the `IndividualAssetReference` field of the GIAI, is a pure integer, with no leading zeros.

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

```
urn:epc:id:sgtin:0652642.800031.400
urn:epc:id:sscc:0652642.0123456789
urn:epc:id:sgln:0652642.12345.400
urn:epc:id:grai:0652642.12345.1234
urn:epc:id:giai:0652642.123456
```

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

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

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

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

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

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

## 4.2 URI Forms for Related Data Types

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

There are several data types that commonly occur in applications that manipulate Electronic Product Codes, which are not themselves Electronic Product Codes but are

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

closely related. This specification provides URI forms for those as well. The general form of the `epc` URN Namespace is

`urn:epc:`*`type`*`:`*`typeSpecificPart`*

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

### 4.2.1 URIs for EPC Tags

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

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

`urn:epc:tag:`*`EncName`*`:`*`EncodingSpecificFields`*

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

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

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

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

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

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

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

Certain bit strings do not correspond to legal encodings. For example, if the most significant bits cannot be recognized as a valid EPC header, the bit-level pattern is not a

legal EPC. For a second example, if the binary value of a field in a tag encoding is greater than the value that can be contained in the number of decimal digits in that field in the URI form, the bit level pattern is not a legal EPC. Nevertheless, software may wish to report such invalid bit-level patterns to users or to other software, and so a representation of invalid bit-level patterns as URIs is provided. The *raw* form of the URI has this general form:

`urn:epc:raw:`*`BitLength`*`.`*`Value`*

where *`BitLength`* is the number of bits in the invalid representation, and *`Value`* is the entire bit-level representation converted to a single decimal number. For example, this bit string:

00000000000000000001001000110100110111101010110110111111011101111

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

`urn:epc:raw:64.20018283527919`

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

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

### 4.2.3 URIs for EPC Patterns

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

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

This pattern refers to any EPC SGTIN Identifier 64-bit tag, whose Filter field is 3, whose Company Prefix is 0652642, whose Item Reference is in the range 1024 = *itemReference* = 2047, and whose Serial Number may be anything at all.

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

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

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

## 4.3 Syntax

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

### 4.3.1 Common Grammar Elements

```
NumericComponent ::= ZeroComponent | NonZeroComponent

ZeroComponent ::= “0”

NonZeroComponent ::= NonZeroDigit Digit*

PaddedNumericComponent ::= Digit+

Digit ::= “0” | NonZeroDigit

NonZeroDigit ::= “1” | “2” | “3” | “4”
               | “5” | “6” | “7” | “8” | “9”
```

### 4.3.2 EPCGID-URI

```
EPCGID-URI ::= “urn:epc:id: gid:” 2*(NumericComponent “.”)
NumericComponent
```

### 4.3.3 SGTIN-URI

```
SGTIN-URI ::= “urn:epc:id:sgtin:” SGTINURIBody

SGTINURIBody ::= 2*(PaddedNumericComponent “.”)
NumericComponent
```

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

### 4.3.4 SSCC-URI

```
SSCC-URI ::= “urn:epc:id:sscc:” SSCCURIBody

SSCCURIBody ::= PaddedNumericComponent “.”
PaddedNumericComponent
```

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

### 4.3.5 SGLN-URI

```
SGLN-qURI ::= “urn:epc:id:sgln:” SGLNURIBody
```

```
SGLNURIBody ::= 2*(PaddedNumericComponent “.”)
NumericComponent
```

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

### 4.3.6 GRAI-URI

```
GRAI-URI ::= “urn:epc:id:grai:” GRAIURIBody
```

```
GRAIURIBody ::= 2*(PaddedNumericComponent “.”)
NumericComponent
```

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

### 4.3.7 GIAI-URI

```
GIAI-URI ::= “urn:epc:id:giai:” GIAIURIBody
```

```
GIAIURIBody ::= PaddedNumericComponent “.”
PaddedNumericComponent
```

The number of characters in the two `PaddedNumericComponent` fields must not exceed 30 (not including any of the dot characters).

### 4.3.8 EPC Tag URI

```
TagURI ::= “urn:epc:tag:” TagURIBody
```

```
TagURIBody ::= GIDTagURIBody | SGTINSGLNGRAITagURIBody |
SSCCTagURIBody | GIAITagURIBody
```

```
GIDTagURIBody ::= GIDTagEncName “:” 2*(NumericComponent
“.”) NumericComponent
```

```
GIDTagEncName ::= “gid-96”
```

```
SGTINSGLNGRAITagURIBody ::= SGTINSGLNGRAITagEncName “:”
NumericComponent “.” 2*(PaddedNumericComponent “.”)
NumericComponent
```

```
SGTINSGLNGRAITagEncName ::= “sgtin-96” | “sgtin-64” |
“sgln-96” | “sgln-64” | ”grai-96” | ”grai-64”
```

```
SSCCGIAITagURIBody ::= SSCCGIAITagEncName “:”
NumericComponent 2*(“.” PaddedNumericComponent)
```

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

```
SSCCGIAITagEncName ::= “sscc-96” | “sscc-64” | “giai-96” |
“giai-64”
```

### 4.3.9 Raw Tag URI

```
RawURI ::= “urn:epc:raw:” RawURIBody

RawURIBody ::= NonZeroComponent “.” NumericComponent
```

### 4.3.10 EPC Pattern URI

```
PatURI ::= “urn:epc:pat:” PatBody

PatBody ::= GIDPatURIBody | SGTINSGLNGRAIPatURIBody |
SSCCGIAIPatURIBody

GIDPatURIBody ::= GIDTagEncName “:” 2*(PatComponent “.”)
PatComponent

SGTINSGLNGRAIPatURIBody ::= SGTINSGLNGRAITagEncName “:”
PatComponent “.” GS1PatBody “.” PatComponent

SSCCGIAIPatURIBody ::= SSCCGIAITagEncName “:” PatComponent
“.” GS1PatBody

GS1PatBody ::= “*.*” | ( PaddedNumericComponent “.”
PatComponent )

PatComponent ::= NumericComponent
               | StarComponent
               | RangeComponent

StarComponent ::= “*”

RangeComponent ::= “[“ NumericComponent “-“
                       NumericComponent “]”
```

For a `RangeComponent` to be legal, the numeric value of the first `NumericComponent` must be less than or equal to the numeric value of the second `NumericComponent`.

### 4.3.11 Summary (non-normative)

The syntax rules above can be summarized informally as follows:

urn:epc:id:gid:*MMM.CCC.SSS*

urn:epc:id:sgtin:*PPP.III.SSS*

urn:epc:id:sscc:*PPP.III*

urn:epc:id:sgln:*PPP.III*

urn:epc:id:grai:*PPP.III.SSS*

urn:epc:id:giai:*PPP.SSS*

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

urn:epc:tag:sgtin-64:*FFF.PPP.III.SSS*
urn:epc:tag:sscc-64:*FFF.PPP.III*
urn:epc:tag:sgln-64:*FFF.PPP.III.SSS*
urn:epc:tag:grai-64:*FFF.PPP.III.SSS*
urn:epc:tag:giai-64:*FFF.PPP.SSS*
urn:epc:tag:gid-96:*MMM.CCC.SSS*
urn:epc:tag:sgtin-96:*FFF.PPP.III.SSS*
urn:epc:tag:sscc-96:*FFF.PPP.III*
urn:epc:tag:sgln-96:*FFF.PPP.III.SSS*
urn:epc:tag:grai-96:*FFF.PPP.III.SSS*
urn:epc:tag:giai-96:*FFF.PPP.SSS*

urn:epc:raw:*LLL.BBB*

urn:epc:pat:sgtin-64:*FFFpat.PPP.IIIpat.SSSpat*
urn:epc:pat:sgtin-64:*FFFpat.*.*.SSSpat*
urn:epc:pat:sscc-64:*FFFpat.PPP.IIIpat*
urn:epc:pat:sscc-64:*FFFpat.*.**
urn:epc:pat:sgln-64:*FFFpat.PPP.IIIpat.SSSpat*
urn:epc:pat:sgln-64:*FFFpat.*.*.SSSpat*
urn:epc:pat:grai-64:*FFFpat.PPP.IIIpat.SSSpat*
urn:epc:pat:grai-64:*FFFpat.*.*.SSSpat*
urn:epc:pat:giai-64:*FFFpat.PPP.SSSpat*
urn:epc:pat:giai-64:*FFFpat.*.**
urn:epc:pat:gid-96:*MMMpat.CCCpat.SSSpat*
urn:epc:pat:sgtin-96:*FFFpat.PPP.IIIpat.SSSpat*
urn:epc:pat:sgtin-96:*FFFpat.*.*.SSSpat*
urn:epc:pat:sscc-96:*FFFpat.PPP.IIIpat*
urn:epc:pat:sscc-96:*FFFpat.*.**
urn:epc:pat:sgln-96:*FFFpat.PPP.IIIpat.SSSpat*
urn:epc:pat:sgln-96:*FFFpat.*.*.SSSpat*
urn:epc:pat:grai-96:*FFFpat.PPP.IIIpat.SSSpat*

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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

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

where

*`MMM`* denotes a General Manager Number

*`CCC`* denotes an Object Class number

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

*`PPP`* denotes an EAN.UCC Company Prefix

*`III`* denotes an SGTIN Item Reference (with Indicator Digit appended to the beginning), an SSCC Shipping Container Serial Number (with the Extension (ED) digit appended at the beginning), a SGLN Location Reference, or a GRAI Asset Type.

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

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

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

*`BBB`* denotes the literal value of an uninterpreted bit sequence converted to decimal

and where all numeric fields are in decimal with no leading zeros (unless the overall value of the field is zero, in which case it is represented with a single `0` character). Exception: the length of *`PPP`* and *`III`* is significant, and leading zeros are used as necessary. The length of *`PPP`* is the length of the company prefix as assigned by EAN or UCC. The length of *`III`* plus the length of *`PPP`* must equal 13 for SGTIN, 17 for SSCC, 12 for GLN, or 12 for GRAI.

# 5 Translation between EPC-URI and Other EPC Representations

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

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

1. Determine the identity type and encoding scheme by finding the row in Table 1 (Section 3.1) that matches the most significant bits of the bit string. If the most significant bits do not match any row of the table, stop: the bit string is invalid and cannot be translated into an EPC-URI. Otherwise, if the encoding scheme is SGTIN-64 or SGTIN-96, proceed to Step 2; if the encoding scheme is SSCC-64 or SSCC-96, proceed to Step 5; if the encoding scheme is SGLN-64 or SGLN-96, proceed to Step 8; if the encoding scheme is GRAI-64 or GRAI-96, proceed to Step 11; if the encoding scheme is GIAI-64 or GIAI-96, proceed to Step 14; if the encoding scheme is GID-96, proceed to Step 17.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

2. Follow the decoding procedure given in Section 3.4.1.2 (for SGTIN-64) or in Section 3.4.2.2 (for SGTIN-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal Item Reference and Indicator $i_1i_2\ldots i_{(13-L)}$, and the Serial Number $S$.
3. Create an EPC-URI by concatenating the following: the string `urn:epc:id:sgtin:`, the Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, the Item Reference and Indicator $i_1i_2\ldots i_{(13-L)}$ (handled similarly), a dot (`.`) character, and the Serial Number $S$ as a decimal integer. The portion corresponding to the Serial Number must have no leading zeros, except where the Serial Number is itself zero in which case the corresponding URI portion must consist of a single zero character.
4. Go to Step 19.
5. Follow the decoding procedure given in Section 3.5.1.2 (for SSCC-64) or in Section 3.5.2.2 (for SSCC-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, and the decimal Serial Reference $s_1s_2\ldots s_{(17-L)}$.
6. Create an EPC-URI by concatenating the following: the string `urn:epc:id:sscc:`, the Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, and the Serial Reference $s_1s_2\ldots s_{(17-L)}$ (handled similarly).
7. Go to Step 19.
8. Follow the decoding procedure given in Section 3.6.1.2 (for SGLN-64) or in Section 3.6.2.2 (for SGLN-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal Location Reference $i_1i_2\ldots i_{(12-L)}$, and the Serial Number $S$.
9. Create an EPC-URI by concatenating the following: the string `urn:epc:id:sgln:`, the Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, the Location Reference $i_1i_2\ldots i_{(12-L)}$ (handled similarly), a dot (`.`) character, and the Serial Number $S$ as a decimal integer. The portion corresponding to the Serial Number must have no leading zeros, except where the Serial Number is itself zero in which case the corresponding URI portion must consist of a single zero character.
10. Go to Step 19.
11. Follow the decoding procedure given in Section 3.7.1.2 (for GRAI-64) or in Section 3.7.2.2 (for GRAI-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal Asset Type $i_1i_2\ldots i_{(12-L)}$, and the Serial Number $S$.
12. Create an EPC-URI by concatenating the following: the string `urn:epc:id:grai:`, the Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, the Asset Type $i_1i_2\ldots i_{(12-L)}$ (handled similarly), a dot (`.`) character, and the Serial Number $S$ as a decimal integer. The portion corresponding to the Serial Number must have no leading zeros, except where the Serial Number is itself zero

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

in which case the corresponding URI portion must consist of a single zero character.

13. Go to Step 19.
14. Follow the decoding procedure given in Section 3.8.1.2 (for GIAI-64) or in Section 3.8.2.2 (for GIAI-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, and the Individual Asset Reference *S*.
15. Create an EPC-URI by concatenating the following: the string `urn:epc:id:giai:`, the Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, and the Individual Asset Reference *S* as a decimal integer. The portion corresponding to the Individual Asset Reference must have no leading zeros, except where the Individual Asset Reference is itself zero in which case the corresponding URI portion must consist of a single zero character.
16. Go to Step 19.
17. Follow the decoding procedure given in Section 3.3.1.2 to obtain the General Manager Number *M*, the Object Class *C*, and the Serial Number *S*.
18. Create an EPC-URI by concatenating the following: the string `urn:epc:id:gid:`, the General Manager Number as a decimal integer, a dot (`.`) character, the Object Class as a decimal integer, a dot (`.`) character, and the Serial Number *S* as a decimal integer. Each decimal number must have no leading zeros, except where the integer is itself zero in which case the corresponding URI portion must consist of a single zero character.
19. The translation is now complete.

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

1. Determine the identity type and encoding scheme by finding the row in Table 1 (Section 3.1) that matches the most significant bits of the bit string. If the encoding scheme is SGTIN-64 or SGTIN-96, proceed to Step 2; if the encoding scheme is SSCC-64 or SSCC-96, proceed to Step 5; if the encoding scheme is SGLN-64 or SGLN-96, proceed to Step 8; if the encoding scheme is GRAI-64 or GRAI-96, proceed to Step 11, if the encoding scheme is GIAI-64 or GIAI-96, proceed to Step 14, if the encoding scheme is GID-96, proceed to Step 17; otherwise, proceed to Step 20.
2. Follow the decoding procedure given in Section 3.4.1.2 (for SGTIN-64) or in Section 3.4.2.2 (for SGTIN-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal Item Reference and Indicator $i_1i_2\ldots i_{(13-L)}$, the Filter Value *F*, and the Serial Number *S*.
3. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`, the encoding scheme (`sgtin-64` or `sgtin-96`), a colon (`:`) character, the Filter Value *F* as a decimal integer, a dot (`.`) character, the Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes

the corresponding ASCII digit character, a dot (`.`) character, the Item Reference and Indicator $i_1 i_2 \ldots i_{(13\text{-L})}$) (handled similarly), a dot (`.`) character, and the Serial Number $S$ as a decimal integer. The portions corresponding to the Filter Value and Serial Number must have no leading zeros, except where the corresponding integer is itself zero in which case a single zero character is used.

4. Go to Step 21.
5. Follow the decoding procedure given in Section 3.5.1.2 (for SSCC-64) or in Section 3.5.2.2 (for SSCC-96) to obtain the decimal Company Prefix $p_1 p_2 \ldots p_L$, and the decimal Serial Reference $i_1 i_2 \ldots s_{(17\text{-L})}$, and the Filter Value $F$.
6. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`, the encoding scheme (`sscc-64` or `sscc-96`), a colon (`:`) character, the Filter Value $F$ as a decimal integer, a dot (`.`) character, the Company Prefix $p_1 p_2 \ldots p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, and the Serial Reference $i_1 i_2 \ldots i_{(17\text{-L})}$ (handled similarly).
7. Go to Step 21.
8. Follow the decoding procedure given in Section 3.6.1.2 (for SGLN-64) or in Section 3.6.2.2 (for SGLN-96) to obtain the decimal Company Prefix $p_1 p_2 \ldots p_L$, the decimal Location Reference $i_1 i_2 \ldots i_{(12\text{-L})}$, the Filter Value $F$, and the Serial Number $S$.
9. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`, the encoding scheme (`sgln-64` or `sgln-96`), a colon (`:`) character, the Filter Value $F$ as a decimal integer, a dot (`.`) character, the Company Prefix $p_1 p_2 \ldots p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, the Location Reference $i_1 i_2 \ldots i_{(12\text{-L})}$ (handled similarly), a dot (`.`) character, and the Serial Number $S$ as a decimal integer. The portions corresponding to the Filter Value and Serial Number must have no leading zeros, except where the corresponding integer is itself zero in which case a single zero character is used.
10. Go to Step 21.
11. Follow the decoding procedure given in Section 3.7.1.2 (for GRAI-64) or in Section 3.7.2.2 (for GRAI-96) to obtain the decimal Company Prefix $p_1 p_2 \ldots p_L$, the decimal Asset Type $i_1 i_2 \ldots i_{(12\text{-L})}$, the Filter Value $F$, and the Serial Number $d_{15} d_2 \ldots d_K$.
12. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`, the encoding scheme (`grai-64` or `grai-96`), a colon (`:`) character, the Filter Value $F$ as a decimal integer, a dot (`.`) character, the Company Prefix $p_1 p_2 \ldots p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, the Asset Type $s_1 s_2 \ldots s_{(12\text{-L})}$ (handled similarly), a dot (`.`) character, and the Serial Number $d_{15} d_2 \ldots d_K$ as a decimal integer. The portions corresponding to the Filter Value

and Serial Number must have no leading zeros, except where the corresponding integer is itself zero in which case a single zero character is used.

13. Got to Step 21.
14. Follow the decoding procedure given in Section 3.8.1.2 (for GIAI-64) or in Section 3.8.2.2 (for GIAI-96) to obtain the decimal Company Prefix $p_1p_2...p_L$, the decimal Individual Asset Reference $s_1s_2\ldots s_J$, and the Filter Value $F$.
15. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:`, the encoding scheme (`giai-64` or `giai-96`), a colon (`:`) character, the Filter Value $F$ as a decimal integer, the Company Prefix $p_1p_2...p_L$ where each digit (including any leading zeros) becomes the corresponding ASCII digit character, a dot (`.`) character, and the Individual Asset Reference $i_1i_2\ldots i_J$ (handled similarly). The portion corresponding to the Filter Value must have no leading zeros, except where the corresponding integer is itself zero in which case a single zero character is used.
16. Go to Step 21.
17. Follow the decoding procedure given in Section 3.3.1.2 to obtain the EPC Manager Number, the Object Class, and the Serial Number.
18. Create an EPC Tag URI by concatenating the following: the string `urn:epc:tag:gid-96:`, the General Manager Number as a decimal number, a dot (`.`) character, the Object Class as a decimal number, a dot (`.`) character, and the Serial Number as a decimal number. Each decimal number must have no leading zeros, except where the integer is itself zero in which case the corresponding URI portion must consist of a single zero character.
19. Go to Step 21.
20. This tag is not a recognized EPC encoding, therefore create an EPC Raw URI by concatenating the following: the string `urn:epc:raw:`, the length of the bit string, a dot (`.`) character, and the value of the bit string considered as a single decimal integer. Both the length and the value must have no leading zeros, except if the value is itself zero in which case a single zero character is used.
21. The translation is now complete.

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

1. If the URI is an SGTIN-URI (`urn:epc:id:sgtin:`), an SSCC-URI (`urn:epc:id:sscc:`), an SGLN-URI (`urn:epc:id:sgln:`), a GRAI-URI (`urn:epc:id:grai:`), a GIAI-URI (`urn:epc:id:giai:`), a GID-URI (`urn:epc:id:gid:`), or an EPC Pattern URI (`urn:epc:pat:`), the URI cannot be translated into a bit-level EPC.
2. If the URI is a Raw Tag URI (`urn:epc:raw:`), create the bit-level EPC by converting the second component of the Raw Tag URI into a binary integer, whose length is equal to the first component of the Raw Tag URI. If the value of

the second component is too large to fit into a binary integer of that size, the URI cannot be translated into a bit-level EPC.

3. If the URI is an EPC Tag URI (`urn:epc:tag:`*`encName`*`:`), parse the URI using the grammar for `TagURI` as given in Section 4.3.8. If the URI cannot be parsed using this grammar, stop: the URI is illegal and cannot be translated into a bit-level EPC. Otherwise, if *`encName`* is `sgtin-96` or `sgtin-64` go to Step 4, if *`encName`* is `sscc-96` or `sscc-64` go to Step 9, if *`encName`* is `sgln-96` or `sgln-64` go to Step 13, if *`encName`* is `grai-96` or `grai-64` go to Step 18, if *`encName`* is `giai-96` or `giai-64` go to Step 23, or if *`encName`* is `gid-96` go to Step 27.
4. Let the URI be written as `urn:epc:tag:`*`encName`*`:`$f_1f_2\ldots f_F.p_1p_2\ldots p_L.i_1i_2\ldots i_{(13-L)}.s_1s_2\ldots s_S$.
5. Interpret $f_1f_2\ldots f_F$ as a decimal integer *F*.
6. Interpret $s_1s_2\ldots s_S$ as a decimal integer *S*.
7. Carry out the encoding procedure defined in Section 3.4.1.1 (SGTIN-64) or Section 3.4.2.1 (SGTIN-96), using $i_1p_1p_2\ldots p_Li_2\ldots i_{(13-L)}0$ as the EAN.UCC GTIN-14 (the trailing zero is a dummy check digit, which is ignored by the encoding procedure), L as the length of the EAN.UCC company prefix, *F* from Step 5 as the Filter Value, and *S* from Step 6 as the Serial Number. If the encoding procedure fails because an input is out of range, or because the procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.
8. Go to Step 32.
9. Let the URI be written as `urn:epc:tag:`*`encName`*`:`$f_1f_2\ldots f_F.p_1p_2\ldots p_L.i_1i_2\ldots i_{(17-L)}$.
10. Interpret $f_1f_2\ldots f_F$ as a decimal integer *F*.
11. Carry out the encoding procedure defined in Section 3.5.1.1 (SSCC-64) or Section 3.5.2.1 (SSCC-96), using $\mathrm{s1P}_1p_2\ldots p_Li_2i_3\ldots i_{(17-L)}0$ as the EAN.UCC SSCC, L as the length of the EAN.UCC company prefix, and *F* from Step 10 as the Filter Value. If the encoding procedure fails because an input is out of range, or because the procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.
12. Go to Step 32.
13. Let the URI be written as `urn:epc:tag:`*`encName`*`:`$f_1f_2\ldots f_F.p_1p_2\ldots p_L.i_1i_2\ldots i_{(12-L)}.s_1s_2\ldots s_S$.
14. Interpret $f_1f_2\ldots f_F$ as a decimal integer *F*.
15. Interpret $s_1s_2\ldots s_S$ as a decimal integer *S*.
16. Carry out the encoding procedure defined in Section 3.6.1.1 (SGLN-64) or Section 3.6.2.1 (SGLN-96), using $p_1p_2\ldots p_Li_1i_2\ldots i_{(12-L)}0$ as the EAN.UCC GLN (the trailing zero is a dummy check digit, which is ignored by the encoding

procedure), L as the length of the EAN.UCC company prefix, *F* from Step 14 as the Filter Value, and *S* from Step 15 as the Serial Number. If the encoding procedure fails because an input is out of range, or because the procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.

17. Go to Step 32.
18. Let the URI be written as
    `urn:epc:tag:`*encName*`:`$f_1f_2\ldots f_F.p_1p_2\ldots p_L.i_1i_2\ldots i_{(12-L)}.s_1s_2\ldots s_S$.
19. Interpret $f_1f_2\ldots f_F$ as a decimal integer *F*.
20. Carry out the encoding procedure defined in Section 3.7.1.1 (GRAI-64) or Section 3.7.2.1 (GRAI-96), using $0p_1p_2\ldots p_Li_1i_2\ldots i_{(12-L)}0s_1s_2\ldots s_S$ as the EAN.UCC GRAI (the second zero is a dummy check digit, which is ignored by the encoding procedure), L as the length of the EAN.UCC company prefix, and *F* from Step 19 as the Filter Value. If the encoding procedure fails because an input is out of range, or because the procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.
21. Go to Step 31.
22. Let the URI be written as
    `urn:epc:tag:`*encName*`:`$f_1f_2\ldots f_F.p_1p_2\ldots p_L.s_1s_2\ldots s_S$.
23. Interpret $f_1f_2\ldots f_F$ as a decimal integer *F*.
24. Carry out the encoding procedure defined in Section 3.8.1.1 (GIAI-64) or Section 3.8.2.1 (GIAI-96), using $p_1p_2\ldots p_Ls_1s_2\ldots s_S$ as the EAN.UCC GIAI, L as the length of the EAN.UCC company prefix, and *F* from Step 24 as the Filter Value. If the encoding procedure fails because an input is out of range, or because the procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.
25. Go to Step 31.
26. Let the URI be written as
    `urn:epc:tag:`*encName*`:`$m_1m_2\ldots m_L.c_1c_2\ldots c_K.s_1s_2\ldots s_S$.
27. Interpret $m_1m_2\ldots m_L$ as a decimal integer *M*.
28. Interpret $c_1c_2\ldots c_K$ as a decimal integer *C*.
29. Interpret $s_1s_2\ldots s_S$ as a decimal integer *S*.
30. Carry out the encoding procedure defined in Section 3.3.1.1 using *M* from Step 28 as the EPC Manager, *C* from Step 29 as the Object Class, and *S* from Step 30 as the Serial Number. If the encoding procedure fails because an input is out of range, or because the procedure indicates a failure, stop: this URI cannot be encoded into an EPC tag.
31. The translation is complete.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# 6 Semantics of EPC Pattern URIs

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

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

First, any EPC Tag URI component `C`*`i`* is said to *match* the corresponding EPC Pattern URI component `P`*`i`* if:

`P`*`i`* is a `NumericComponent`, and `C`*`i`* is equal to `P`*`i`*; or

`P`*`i`* is a `PaddedNumericComponent`, and `C`*`i`* is equal to `P`*`i`* both in numeric value as well as in length; or

`P`*`i`* is a `RangeComponent [`*`lo`*`-`*`hi`*`]`, and *`lo`* = `C`*`i`* = *`hi`*; or

`P`*`i`* is a `StarComponent` (and `C`*`i`* is anything at all)

Then the EPC Tag URI is a member of the set denoted by the EPC Pattern URI if and only if `Ci` matches `Pi` for all 1 = *`i`* = *`n`*.

# 7 Background Information

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

### EPC Tag Data Standards Group

| | |
|---|---|
| Bud Babcock | Procter & Gamble |
| Jason Carney | Ahold |

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

| | |
|---|---|
| Hal Charych | Symbol |
| Chris Cummins | Uniform Code Council |
| Falk Gernot | Metro |
| Yuichiro Hanawa | Mitsui, USA |
| Mark Harrison | Cambridge Auto-ID Lab |
| Larry Hilgert | Pepsico (Quaker) |
| André Frank | Sara Lee/DE |
| Jonathan Loretto | Cap Gemini Ernst & Young |
| Ron Moser | Wal-Mart |
| Don Mowery | Nestle |
| Bob Mytkowicz | Gillette |
| Doug Naal | Kraft |
| Juli Nackers | Kimberly-Clark |
| Robert Nonneman | UPS |
| Richard Probst | Nominum |
| Steve Rehling | Procter & Gamble |
| Rick Schendel | Target |
| Sylvia Stein | EAN Netherlands |
| Richard Swan | T3C1 |
| Michael Szafranski | Kraft |
| Nancy Tai | Georgia Pacific |
| Ken Traub | Connecterra, Inc. |

### URI Representation Group

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

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

The following papers capture the contributions of these individuals:

Engels, D., Foley, J., Waldrop, J., Sarma, S. and Brock, D., "The Networked Physical World: An Automated Identification Architecture" Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD01), 76-77, 2001.

Brock, David. "The Electronic Product Code (EPC), A Naming Scheme for Physical Objects", 2001.

Brock, David. "The Compact Electronic Product Code; A 64-bit Representation of the Electronic Product Code", 2001.http://www.autoidcenter.org/publishedresearch/MIT-AUTOID-WH-002.pdf

# 8 References

"General EAN.UCC Specifications." Version 5.0, EAN International and the Uniform Code Council, Inc™, January 2004.

[MIT-TR009] D. Engels, "The Use of the Electronic Product Code™," MIT Auto-ID Center Technical Report MIT-TR007, February 2003, http://www.autoidcenter.org/publishedresearch/mit-autoid-tr009.pdf.

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# 9 Appendix A: Encoding Scheme Summary Tables

## SGTIN Summary

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

| SGTIN-96 | Header | Filter Value | Partition | Company Prefix | Item Reference | Serial Number |
|---|---|---|---|---|---|---|
| | 8 | 3 | 3 | 20-40 | 24 - 4 | 38 |
| | 0011 0000 (Binary value) | 8 (Decimal capacity) | 8 (Decimal capacity) | 999,999 – 999,999,999,999 (Decimal capacity**) | 9,999,999 – 9 (Decimal capacity**) | 274,877,906,943 (Decimal capacity) |

**Filter Values (Non-normative)**

| Type | Binary Value |
|---|---|
| Other | xxx |
| Item | xxx |
| Inner Pack | xxx |
| Case | xxx |
| Load/Pallet | xxx |
| Reserved | xxx |

**SGTIN Partition Table**

| Partition Value | Company Prefix | | Item Reference and Indicator Digit | |
|---|---|---|---|---|
| | Bits | Digits | Bits | Digit |
| 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 |

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## SSCC Summary

| SSCC-64 | Header | Filter Value | Company Prefix Index | Serial Reference |
|---|---|---|---|---|
| | 8 | 3 | 14 | 39 |
| | 0000 1000 (Binary value) | 8 (Decimal capacity) | 16,383 (Decimal capacity) | 99,999 - 99,999,999,999 (Decimal capacity*) |

| SSCC-96 | Header | Filter Value | Partition | Company Prefix | Serial Reference | Unallocated |
|---|---|---|---|---|---|---|
| | 8 | 3 | 3 | 20-40 | 37-17 | 25 |
| | 0011 0001 (Binary value) | 8 (Decimal capacity) | 8 (Decimal capacity) | 999,999 – 999,999,999,999 (Decimal capacity**) | 99,999,999,999 – 99,999 (Decimal capacity**) | [Not Used] |

| Filter Values (Non-normative) | | SSCC Partition Table | | | | |
|---|---|---|---|---|---|---|
| **Type** | **Binary Value** | **Partition Value** | **Company Prefix** | | **Serial Reference and extension digit** | |
| Other | xxx | | **Bits** | **Digits** | **Bits** | **Digits** |
| Case | xxx | 0 | 40 | 12 | 14 | 4 |
| Load/Pallet | xxx | 1 | 37 | 11 | 17 | 5 |
| | | 2 | 34 | 10 | 20 | 6 |
| | | 3 | 30 | 9 | 24 | 7 |
| Reserved | xxx | 4 | 27 | 8 | 27 | 8 |
| | | 5 | 24 | 7 | 30 | 9 |
| | | 6 | 20 | 6 | 34 | 10 |

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## SGLN Summary

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

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

**Filter Values (Non-normative)**

| Type | Binary Value |
|---|---|
| Other | xxx |
| Physical Location | xxx |
| Reserved | xxx |

**SGLN Partition Table**

| Partition Value | Company Prefix | | Location Reference | |
|---|---|---|---|---|
| | Bits | Digits | Bits | Digit |
| 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 |

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## GRAI Summary

| GRAI-64 | Header | Filter Value | Company Prefix Index | Asset Type | Serial Number |
|---|---|---|---|---|---|
| | 8 | 3 | 14 | 20 | 19 |
| | 0000 1010 (Binary value) | 8 (Decimal capacity) | 16,383 (Decimal capacity) | 9,999,999 - 9 (Decimal capacity*) | 524,288 (Decimal capacity) |

| GRAI-96 | Header | Filter Value | Partition | Company Prefix | Asset Type | Serial Number |
|---|---|---|---|---|---|---|
| | 8 | 3 | 3 | 20-40 | 24 - 4 | 38 |
| | 0011 0011 (Binary value) | 8 (Decimal capacity) | 8 (Decimal capacity) | 999,999 – 999,999,999,999 (Decimal capacity**) | 9,999,999 – 9 (Decimal capacity**) | 274,877,906,943 (Decimal capacity) |

**Filter Values (Non-normative)**

| Type | Binary Value |
|---|---|
| TBD | |
| Reserved | xxx |

**GRAI Partition Table**

| Partition Value | Company Prefix | | Asset Type | |
|---|---|---|---|---|
| | Bits | Digits | Bits | Digit |
| 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 |

*Capacity of Asset Type field varies with Company Prefix.

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

## GIAI Summary

| GIAI-64 | Header | Filter Value | Company Prefix Index | Individual Asset Reference |
|---|---|---|---|---|
| | 8 | 3 | 14 | 39 |
| | 0000 1011<br>(Binary value) | 8<br>(Decimal capacity) | 16,383<br>(Decimal capacity) | 549,755,813,888<br>(Decimal capacity) |

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

**Filter Values**

**(To be confirmed)**

| Type | Binary Value |
|---|---|
| TBD | |
| Reserved | xxx |

**GIAI Partition Table**

| Partition Value | Company Prefix | | Individual Asset Reference | |
|---|---|---|---|---|
| | **Bits** | **Digits** | **Bits** | **Digits** |
| 0 | 40 | 12 | 42 | 12 |
| 1 | 37 | 11 | 45 | 13 |
| 2 | 34 | 10 | 48 | 14 |
| 3 | 30 | 9 | 52 | 15 |
| 4 | 27 | 8 | 55 | 16 |
| 5 | 24 | 7 | 58 | 17 |
| 6 | 20 | 6 | 62 | 18 |

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# 10 Appendix B: EPC Header Values and Tag Identity Lengths

With regards to tag identity lengths and EPC Header values: In the decoding process of a single tag: Having knowledge of the identifier length during the signal decoding process of the reader enables the reader to know when to stop trying to decode bit values. Knowing when to stop enables the readers to be more efficient in reading speed. For example, if the same Header value is used at 64 and 96 bits, the reader, upon finding that header value, must try to decode 96 bits. After decoding 96 bits, the reader must check the CRC (Cyclic Redundancy Check error check code) against both the 64-bit and 96-bit numbers it has decoded. If both error checks fail, the numbers are thrown away and the tag reread. If one of the numbers passes the error check, then that is reported as the valid number. Note that there is a non-zero, i.e., greater than zero but very small, probability that an erroneous number can be reported in this process. If both numbers pass the error check, then there is a problem. Note that there is a small probability that both a 64 bit

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

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

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

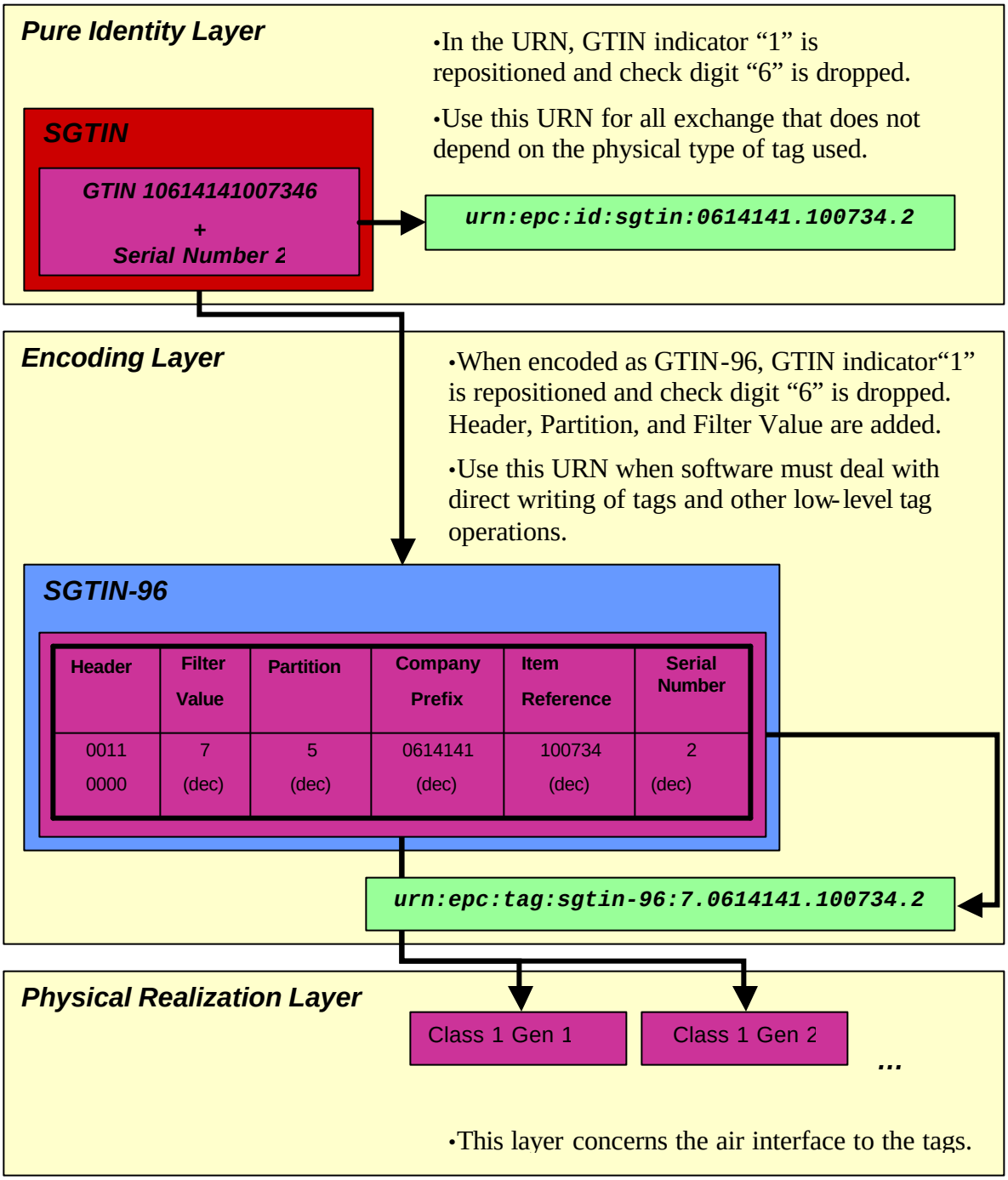
communicating tags are of the same length and the reader knows what that length is a priori. For efficiency reasons, a single selection for a length is preferred, but two can be workable. More than two becomes very inefficient.

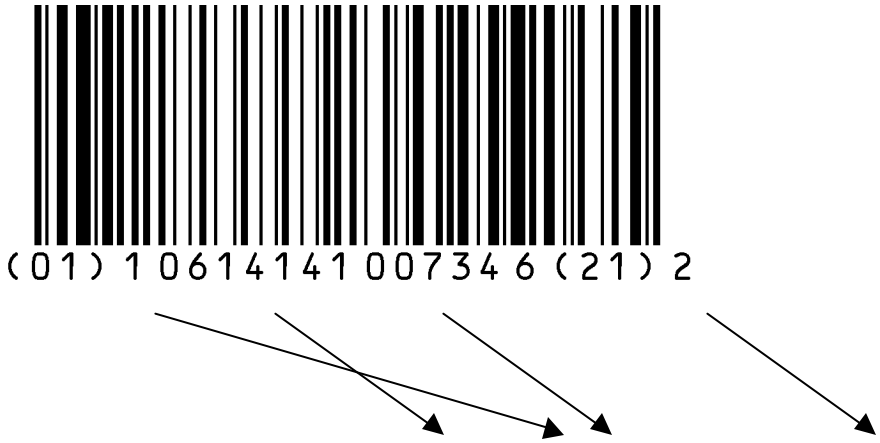
The net effect of encoding the length within the Header is to reduce the probabilities of error in the decoding process and to increase the efficiency of the identification process.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

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

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






| | Header | Filter Value | Partition | Company Prefix | Item Reference | Serial Number |
|---|---|---|---|---|---|---|
| SGTIN-96 | 8 bits | 3 bits | 3 bits | 24 bits | 20 bits | 38 bits |
| | 0011 0000 (Binary value) | 7 (Decimal value) | 5 (Decimal value) | 0614141 (Decimal value) | 100734 (Decimal value) | 2 (Decimal value) |

?? (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 is currently not defined, so 7 is a notional value.

?? 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.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# 12 Appendix D: Binary Digit Capacity Tables

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# 13 Appendix E: List of Abbreviations

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

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# 14 Appendix F: General EAN.UCC Specifications

## Section 3.0 Definition of Element Strings and Section 3.7 EPCglobal Tag Data Standard

This section provides EAN.UCC approval of the EPCglobal? Tag Data Standard V1.1, Rev. 1.23 with the following EAN.UCC Application Identifier definition restrictions:

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

- ?? 64-bit tag application is limited to 16,383 EAN.UCC Company Prefixes and therefore EAN.UCC EPCglobal implementation strategies will focus on tag capacity that can accommodate all EAN.UCC member companies. The 64-bit tag will be approved for use by EAN.UCC member companies with the restrictions that follow:
- ?? **AI (00) SSCC** (no restrictions)
- ?? **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is restricted to permit assignment of 33,554,431 numeric-only serial numbers.
- ?? **AI (41n) GLN + AI (21) Serial Number:** The Tag Data Standard V1.1 R1.23 is approved with a complete restriction on GLN serialization because this question has not been resolved by GSMP at this time.
- ?? **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier definition is restricted to permit assignment of 524,288 numeric-only serial numbers and the serial number element is mandatory.
- ?? **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier definition is restricted to permit assignment of 549,755,813,888 numeric-only serial numbers.

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

- ?? **AI (00) SSCC** (no restrictions)
- ?? **AI (01) GTIN + AI (21) Serial Number:** The Section 3.6.13 Serial Number definition is restricted to permit assignment of 274,877,906,943 numeric-only serial numbers)
- ?? **AI (41n) GLN + AI (21) Serial Number:** The Tag Data Standard V1.1 R1.23 is approved with a complete restriction on GLN serialization because this question has not been resolved by GSMP at this time.
- ?? **AI (8003) GRAI Serial Number:** The Section 3.6.49 Global Returnable Asset Identifier definition is restricted to permit assignment of 274,877,906,943 numeric-only serial numbers and the serial number element is mandatory.
- ?? **AI (8004) GIAI Serial Number:** The Section 3.6.50 Global Individual Asset Identifier definition is restricted to permit assignment of 4,611,686,018,427,387,904 numeric-only serial numbers.

*Copyright ©2004 EPCglobal®, All Rights Reserved.* 

# Exhibit SWEENEY01

screenshot-library.uakron.edu-2024.08.25-15_26_01
https://library.uakron.edu/search~S24?/.b2868534/.b2868534/1%2C1%2C1%2CB/marc~b2868534
25.08.2024



Start Over Regular Display

```
LEADER 00000cam  22000004a 4500
001    58752830
003    OCoLC
005    20071115000000.0
008    050201s2005    inua     b    001 0 eng
010    2005921609
015    GBA484218|2bnb
016 7  013046272|2Uk
018    57063835
020    076457910X (pbk. : alk. paper)
020    9780764579103 (pbk. : alk. paper)
035    (OCoLC)58752830|z(OCoLC)57063835
035    2144798
040    DLC|cDLC|dUKM|dBAKER|dJRZ|dSPP|dVP@|dBTCTA|dYDXCP
042    pcc
049    AKRS
050 00 TK6553|b.S87 2005
050 00 TK6553|b.S87 2005
082 00 621.384|222
082 04 658.787|222
100 1  Sweeney, Patrick J.,|bII
245 10 RFID for dummies /|cPatrick J. Sweeney
250    1st ed
260    Indianapolis, IN :|bWiley Pub., Inc.,|c2005
300    xviii, 388 p. :|bill. ;|c24 cm
440  0 --For dummies
504    Includes bibliographical references (p. [343]-348) and
       index
650  0 Radio frequency identification systems
915    YBP/PromptCat
994    92|bAKR
```

| LOC | CALL # | STATUS |
|---|---|---|
| Bierce | TK6553 .S87 2005 | DUE 09-20-24 |

Report a Problem

screenshot-library.uakron.edu-2024.08.25-15_25_07
https://library.uakron.edu/record=b2868534~S24
25.08.2024

University of Akron Libraries Catalog

UA Home | University Libraries | Contact the Library | My Library

(Search History)

RECORD # | b2868534 | View Entire Collection | Search

Start Over | Request | MARC Display

Book

| | |
|---|---|
| Title | **RFID for dummies / Patrick J. Sweeney** |
| Author | Sweeney, Patrick J., II |
| Imprint | Indianapolis, IN : Wiley Pub., Inc., 2005 |
| Edition | 1st ed |

| LOC | CALL # | STATUS |
|---|---|---|
| Bierce | TK6553 .S87 2005 | DUE 09-20-24 |

OhioLINK | SearchOhio

| | |
|---|---|
| Call # | TK6553 .S87 2005 |
| Phys Descr | xviii, 388 p. : ill. ; 24 cm |
| Series | --For dummies |
| Bibliog | Includes bibliographical references (p. [343]-348) and index |
| Subject | Radio frequency identification systems |
| OCLC # | 58752830<br>(OCoLC)58752830 (OCoLC)57063835<br>2144798 |
| Other Number | 013046272 Uk |
| ISBN | 076457910X (pbk. : alk. paper)<br>9780764579103 (pbk. : alk. paper) |
| LC CARD # | 2005921609 |

Permanent link to this record:
https://library.uakron.edu:443/record=b2868534~S24

Report a Problem

# Exhibit SWEENEY02

Get up to speed on Radio I ... fication

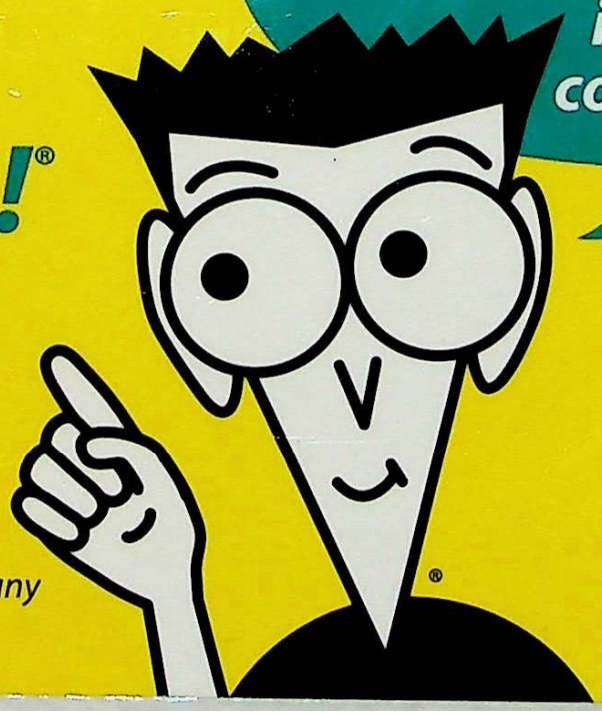
UNIVERSITY OF AKRON
310029 029 31525 0

# RFID FOR DUMMIES®

Get ready to implement this revolutionary inventory control tool

A Reference for the Rest of Us!®

FREE eTips at dummies.com®



**Patrick J. Sweeney II**
*CEO of ODIN Technologies, an RFID company*

#1

Bestselling Computer Book Series

# RFID

FOR DUMMIES

Science
TK
6553
S87
2005

Sweeney

## Discover how RFID can save your business money

## See what goes into an RFID set-up, choose your equipment, and test your system

Is the rush to implement RFID causing you sleeplessnights? Take heart. As more and more retailers require their suppliers to get on the RFID bandwagon, this book can save the day. Even if you're challenged and skipped physics class, you'll discover how RFID works and how to set up and deploy your network — just in time.

**Patrick J. Sweeney II** heads ODIN Technologies, an RFID integration and software development company. He learned the technology at MIT and is considered an industry expert.

## Discover how to:

- *Plan your RFID implementation*
- *Evaluate and choose system components*
- *Compare types of RFID tags*
- *Assess your site*
- *Set up and test readers*
- *Administer and maintain your system*

THE DUMMIES WAY®

- *Explanations in plain English*
- *"Get in, get out" information*
- *Icons and other navigational aids*
- *Tear-out cheat sheet*
- *Top ten lists*
- *A dash of humor and fun*

## Get smart! @www.dummies.com

- *Find listings of all our books*
- *Choose from many different subject categories*
- *Sign up for eTips at etips.dummies.com*

ISBN 0-7645-7910-X

$24.99 US
$35.99 CN
£15.99 UK

Technology/
Telecommunications

52499
9 780764 579103

For Dummies®
A Branded Imprint of
WILEY

THE UNIVERSITY OF AKRON LIBRARIES

# RFID FOR DUMMIES®

# RFID FOR DUMMIES®

by Patrick J. Sweeney II

WILEY
Wiley Publishing, Inc.

**RFID For Dummies®**
Published by
**Wiley Publishing, Inc.**
111 River Street
Hoboken, NJ 07030-5774
www.wiley.com

Copyright © 2005 by Wiley Publishing, Inc., Indianapolis, Indiana

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate percopy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at http://www.wiley.com/go/permissions.

**Trademarks:** Wiley, the Wiley Publishing logo, For Dummies, the Dummies Man logo, A Reference for the Rest of Us!, The Dummies Way, Dummies Daily, The Fun and Easy Way, Dummies.com, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

**LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.**

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 800-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

For technical support, please visit www.wiley.com/techsupport.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2005921609

ISBN: 0-7645-7910-X

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2

1O/RU/QT/QV/IN

WILEY

# *About the Author*

As you may have guessed by the dangling participles and misused gerunds, this is the first book by **Patrick J. Sweeney II** (despite Amazon's link to books on gynecology by an author of the same name). When not negotiating with his editor to push back book deadlines, he leads ODIN technologies as President and CEO.

ODIN technologies is a global RFID software and services company focusing on RFID infrastructure. Mr. Sweeney is well recognized as a visionary in the RFID industry with several RFID patents in various stages of approval. He has appeared in such publications as *CIO Magazine, The Washington Post, Fortune* magazine, *Internet Week,* and many others. He has been interviewed by ABC news and CNN, among others, and is a frequent speaker worldwide on all topics relating to RFID. He is also an active member of several standards bodies and regulatory groups helping to shape the evolution of the RFID industry.

Mr. Sweeney is a second-generation IT professional; his father was one of the first employees at Electronic Data Systems (EDS), where "Pops" entertained him and his brother on weekends by teaching them to read punch cards and other useful skills. Mr. Sweeney took that genetic proclivity toward data centers and started a successful, secure managed hosting company in the late 1990s, which he later sold. His brother took that same early training and started XS Speed Choppers, making custom motorcycles — go figure.

Mr. Sweeney finished second in the 1996 Olympic trials in the single scull, is an avid outdoorsman, enjoys helping other entrepreneurs, and is passionate about various Irish causes. He is a board member of Trinity College business school in Dublin, Ireland, and an Alumni Board member at the Darden School of Business at the University of Virginia. He graduated from Darden and received a Bachelor of Science degree from the University of New Hampshire. He is blessed with a great family—wife Christen, daughter Shannon, son P.J., and three dogs. They live in Middleburg, Virginia, in a house full of useless RFID gadgets.

## Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at www.dummies.com/register/.

Some of the people who helped bring this book to market include the following:

***Acquisitions, Editorial, and Media Development***

**Project Editor:** Rebecca Huehls

**Acquisitions Editor:** Melody Layne

**Copy Editor:** Andy Hollandbeck

**Technical Editor:** Christopher Bratten

**Editorial Manager:** Leah P. Cameron

**Media Development Manager:** Laura VanWinkle

**Media Development Supervisor:** Richard Graves

**Editorial Assistant:** Amanda Foxworth

**Cartoons:** Rich Tennant, www.the5thwave.com

***Composition Services***

**Project Coordinators:** Adrienne Martinez, Emily Wichlinski

**Layout and Graphics:** Carl Byers, Andrea Dahl, Lauren Goddard, Denny Hager, Joyce Haughey, Lynsey Osborn, Melanee Prendergast

**Proofreaders:** Laura Albert, Leeann Harney, Jessica Kramer, Linda Morris, Carl William Pierce

**Indexer:** TECHBOOKS Production Services

***Special Help:*** Kim Darosett, Teresa Artman

---

**Publishing and Editorial for Technology Dummies**

**Richard Swadley,** Vice President and Executive Group Publisher

**Andy Cummings,** Vice President and Publisher

**Mary Bednarek,** Executive Acquisitions Director

**Mary C. Corder,** Editorial Director

**Publishing for Consumer Dummies**

**Diane Graves Steele,** Vice President and Publisher

**Joyce Pepple,** Acquisitions Director

**Composition Services**

**Gerry Fahey,** Vice President of Production Services

**Debbie Stailey,** Director of Composition Services

# Contents at a Glance


*Introduction* ..... 1

*Part I: Now That You Can Spell RFID, Here's the Rest of the Story* ..... 7

Chapter 1: Taking the Mystery out of RFID ..... 9
Chapter 2: Auto-ID Technologies: Why RFID Is King of the Hill ..... 31
Chapter 3: Making Basic Decisions about Your RFID System ..... 55

*Part II: Ride the Electromagnetic Wave: The Physics of RFID* ..... 75

Chapter 4: What Makes Up an RFID Network ..... 77
Chapter 5: Understanding How Technology Becomes a Working System ..... 87
Chapter 6: Seeing Different RFID Systems at Work ..... 103

*Part III: Fitting an RFID Application into Your World* ..... 117

Chapter 7: Seeing the Invisible: The Site Assessment ..... 119
Chapter 8: Testing One, Two, Three: Developing Your Own Lab ..... 139
Chapter 9: Tag, You're It: Testing for Best Tag Design and Placement ..... 159
Chapter 10: Hooked on Phonics: Reader Testing, Selection, and Installation ..... 181
Chapter 11: Middle Where? It's Not Just about the Readers ..... 205

*Part IV: Raising the Beams for Your Network* ..... 219

Chapter 12: From Pilot to Admiral: Deploying RFID Successfully ..... 221
Chapter 13: Getting Set to Administer and Maintain Your System ..... 233
Chapter 14: Ping-pong, the Tags Are Gone: How to Monitor Your RFID Network ..... 249

*Part V: How to Speak Bean Counter* ..... 269

Chapter 15: Making the Business Case ..... 271
Chapter 16: Fitting RFID into Strategic Plans ..... 289
Chapter 17: What to Look for When Considering Outsourcing ..... 307

*Part VI: The Part of Tens* .....333
Chapter 18: Ten (Or So) Equipment Vendors .....335
Chapter 19: Ten Web Sites for Information on RFID .....343
Chapter 20: Ten Tips from the Experts .....349
Chapter 21: Ten (Or So) RFID Standards and Protocols .....357

*Appendix: Glossary of Electrical, Magnetic, and Other Scientific Terms* .....363

*Index* .....373

# Table of Contents


*Introduction* ..... *1*

About This Book ..... 1
Who This Book Is For ..... 1
You Don't Need a Slide Rule and Pocket Protector to Use This Book ..... 2
How This Book Is Organized ..... 2
Part I: Now That You Can Spell RFID, Here's the Rest of the Story ..... 3
Part II: Ride the Electromagnetic Wave: The Physics of RFID ..... 3
Part III: Fitting an RFID Application into Your World ..... 3
Part IV: Raising the Beams for Your Network ..... 4
Part V: How to Speak Bean Counter ..... 4
Part VI: The Part of Tens ..... 4
Icons Used in This Book ..... 5

*Part I: Now That You Can Spell RFID, Here's the Rest of the Story* ..... *7*

**Chapter 1: Taking the Mystery out of RFID** ..... **9**

What Is RFID? ..... 9
The origins of RFID in inventory tracking ..... 10
Tracking goods with EPC codes ..... 10
Sizing Up the Benefits of RFID ..... 11
Tracking individual items with serialized data ..... 12
Reducing human intervention ..... 13
Moving more goods through the supply chain ..... 14
Capturing information in real time ..... 14
Increasing security ..... 15
Mandates, Womendates, Blind Dates — Forcing Efficiency ..... 16
What are the major mandates? ..... 16
Responding to the mandates ..... 17
Calling All Physicists! Calling All Physicists! ..... 18
Finding a physics expert ..... 19
The basic physics of RFID ..... 19
Finding Success with Four Ps in a Pod ..... 22
Planning ..... 22
Physics ..... 24
Pilot ..... 26
Production ..... 27
A Ride in the Time Machine ..... 28

**Chapter 2: Auto-ID Technologies: Why RFID Is King of the Hill .....31**

Planning an Auto-ID Strategy for the Times .....32
Comparing the major players in Auto-ID: Bar codes, contact memory, and RFID .....34
Crafting an Auto-ID strategy for your business (Or, why RFID is the wave of the future) .....41
To EPC or Not to Be: Unraveling the Words, Words, Words of the Electronic Product Code .....44
How EPC is different from UPC .....45
Why an EPC RFID tag doesn't contain more information .....47
How the EPC works .....48
How the EPC prepared for the future, and who oversees that .....52
Addressing Privacy Concerns .....53

**Chapter 3: Making Basic Decisions about Your RFID System .....55**

Midas Touch Points: Where RFID Impacts Your Organization .....56
Outlining how RFID affects your business processes .....57
Determining how RFID will affect your facility .....60
Evaluating your technical needs .....61
What's the Frequency, Kenneth? .....64
Understanding the difference between licensed and unlicensed frequencies .....65
Examining the most common frequencies in RFID .....65
Frequencies, power, and countries .....67
Beyond UHF: Looking toward the future .....68
Speed, Accuracy, or Distance — Pick Two .....69
Designing for the right read distance .....70
Reads — tell me how fast and how many .....71
Reading multiple tags at once — accuracy considerations .....72
Now What about the Tags and Objects? .....73

***Part II: Ride the Electro-magnetic Wave: The Physics of RFID .....75***

**Chapter 4: What Makes Up an RFID Network .....77**

Elements of a Basic RFID System .....77
Everything starts with the tag .....79
Antennas send and receive radio waves .....79
Readers tell the antennas what to do .....80
The middleware transforms the system into a network of objects .....80
Time to Make Some Waves — Electromagnetic Waves .....81
Frequency is a measurement .....83
History may repeat itself, but virginity comes only once .....84
Fields: Electrical and magnetic, near and far .....84
Creating resonance between the antennas and the field .....85

**Chapter 5: Understanding How Technology Becomes a Working System ..... 87**

Anatomy of a Passive Tag: Understanding How It Works and Choosing the Right One ..... 88
How do tags receive and transmit information? ..... 88
How does a tag antenna work, and how do you choose among the different kinds? ..... 90
How does the integrated circuit affect performance? ..... 92
Some tag examples for the geek in you ..... 94
Tracking the Tags with a Reader ..... 95
Holler back, young 'un — Transmitting and receiving signals ..... 95
The DSP chip: Examining the brain of a reader ..... 96
Ring around the dipole and other bad antenna stories ..... 98
Air in Her Face — Blowing Sweet Nothings ..... 100

**Chapter 6: Seeing Different RFID Systems at Work ..... 103**

Setting Up RFID Interrogation Zones ..... 103
Coming and going — Reading at a dock door ..... 104
Your gateway to good reads — Other portals ..... 106
Keep on rollin' — Setting up RFID at a conveyor ..... 108
That's a wrap — Interrogating at a shrink-wrap station ..... 109
One at a time — Reading objects on a shelf ..... 110
From Ski Resorts to Airlines: Applying RFID in the Real World ..... 112
Ski resorts ..... 112
Law enforcement ..... 113
Pharmaceuticals ..... 113
Additional business applications ..... 114

***Part III: Fitting an RFID Application into Your World ..... 117***

**Chapter 7: Seeing the Invisible: The Site Assessment ..... 119**

Planning for Your Site Assessment ..... 120
Getting the right test equipment ..... 122
Setting up for RF testing ..... 124
Measuring for AEN during Normal Operations (And Beyond) ..... 126
Testing key points around the warehouse ..... 127
I've been a wild rover for many's a year ..... 127
I don't hear anything; time to make my own noise ..... 129
Solving interference problems ..... 130
Testing to Plan Your RFID Installation ..... 130
Gathering your equipment ..... 131
Comparing the perfect signal to the actual signal ..... 132
Setting up the equipment ..... 133
Conducting the test ..... 134
Putting your results to use ..... 136

**Chapter 8: Testing One, Two, Three: Developing Your Own Lab ....139**

To Lab or Not to Lab ....140
Beyond a Swanky White Lab Coat: The Tools You Need for Successful Testing ....141
Setting Up Your Lab ....142
X-ray marks the spot: Find the perfect location ....143
Physics eye for the lab guy: Design the physical layout ....145
Set up the test equipment ....148
Build specific test equipment ....151
Develop and implement standardized test procedures ....153

**Chapter 9: Tag, You're It: Testing for Best Tag Design and Placement ....159**

Ready, Set, Test! ....160
Looking at the Material Composition of the Items You're Tagging ....162
Examining RF transparent, reflecting, and absorbing materials ....163
Using the RF friendliness pyramid to understand the optimal spot for testing ....164
Choosing a Tag to Test ....166
Testing Tags in an Applications Test Facility ....168
Setting up the testing environment ....170
Carrying out the test ....170
Frequency Response Characterization: Testing Tags with Physics ....171
Encoding and Applying Tags ....174
Tag and ship ....174
Inline production application ....176
The Secrets of Read Success ....177
Avoiding cross talk ....177
Ensuring high-speed reads ....178
Executing full pallet reads ....178

**Chapter 10: Hooked on Phonics: Reader Testing, Selection, and Installation ....181**

Choosing a Hand-held, Mobile, or Fixed-location Reader ....182
Reading between the Lines: Critical Buying Criteria ....183
Consider all the costs involved ....184
Test reader performance ....186
Assess connectivity ....192
Evaluate how well the reader can be fine-tuned ....196
Installing a Reader and Antennas ....201
Mount the reader ....202
Mount and connect the antennas ....203
Power up the reader ....203
Test the interrogation zone for RF path loss ....204

**Chapter 11: Middle Where? It's Not Just about the Readers .....205**

Filter, Smooth, Route: Understanding What You Need Middleware to Do .....206
Exploring Middleware Vendors and Their Offerings .....208
Piecing Together a Middleware Architecture .....210
No more tiers: Grasping the many levels of a middleware architecture .....211
Taking stock of existing investments and skills .....213
Early bird or late bloomer? Prioritizing your middleware needs .....215
Getting the Most from Your RFID Middleware .....216

***Part IV: Raising the Beams for Your Network .....219***

**Chapter 12: From Pilot to Admiral: Deploying RFID Successfully .....221**

Creating a Pilot Project Plan .....222
Start with your major tasks and timeline .....223
Deliverable tracker .....224
There's always an issue with you: Tracking and resolving problems .....225
There is no I in team (but there is an M and an E) .....226
Factors for a Successful Pilot Test .....227
Clearly defined scope .....227
Experienced project manager .....228
Key executive support .....228
User involvement .....228
Specific measurements and metrics .....229
Risk mitigation .....229
Phased approach .....229
Moving from Pilot to Production .....231
Getting the most of your pilot data: The project debrief .....231
Tips for a successful production system .....232

**Chapter 13: Getting Set to Administer and Maintain Your System .....233**

Configuring and Setting Up Tag Readers .....234
Before you begin .....234
Stepping through a reader setup .....235
Creating configuration classes .....236
Getting the Digits .....238
A simple hierarchy for assigning numbers .....238
Allocating unique numbers across many lines and locations .....239

Applying Tags to Objects ....240
Applying tags without breaking them ....240
North by northwest as the corrugation travels: Orienting tags on objects ....241
Sending Objects through Your Business ....242
Lining up tags and readers ....242
Just like the neonatal ward: Handle with care ....243
School's in Session — Training Your Staff ....244
Starting readers manually ....244
Identifying and responding to missed reads ....245
Reinforcing processes versus changing them ....246
Explaining how RFID affects employees ....247

**Chapter 14: Ping-pong, the Tags Are Gone: How to Monitor Your RFID Network ....249**

Why Monitor an RFID Station? ....250
Setting up Two Types of Monitoring ....251
Checking That a Reader Is Active ....251
Choosing the right method ....252
A simple human interface: Enabling operators to monitor the system ....252
Measuring and Interpreting System Behavior ....255
Building a statistical monitoring approach ....255
Breaking data into time intervals ....257
Measure 1: The average tag traffic volume (ATTV) ....259
Measure 2: Read errors to total reads (RETR) ....261
Measure 3: Read error change rates (RECR) ....262
Measure 4: Actual versus predicted traffic rate (APTR) ....262
Measure 5: Mean time between failure (MTBF) ....263
Monitoring as you expand your RFID network ....265
Setting up a monitoring system ....265

***Part V: How to Speak Bean Counter ....269***

**Chapter 15: Making the Business Case ....271**

Finding the First-Round Draft Picks for Your RFID Team ....271
A Game Plan Is More Than Xs and Os — Use a Proven Methodology ....274
Step 1. Refine the process and conduct team training ....275
Step 2. Determine scope and assumptions ....276
Step 3. Determine drivers, strategies, and enablers ....277
Step 4. Identify and assess business processes and interfaces ....279
Step 5. Identify complementary or competing business initiatives ....280
Step 6. Identify strategic and economic benefits ....281
Step 7. Develop investment requirements ....284
Step 8. Develop an implementation road map ....285
Step 9. Communicate the business case ....286

**Chapter 16: Fitting RFID into Strategic Plans .....289**

Just in Time to Justify: Overcoming Skepticism with Strategic Thinking .....290
Calculating ROI — A Tactical Approach to RFID .....291
Cha-ching! Finding ways to save with RFID .....292
Tallying up the estimated costs .....300
Putting together a costs/benefits analysis .....303
ROI as a tool for strategic expansion .....303
Tag and You're It: RFID as a Competitive Strategy .....304

**Chapter 17: What to Look for When Considering Outsourcing .....307**

Why Outsource Your RFID Network? .....308
Identifying and Avoiding the Risks .....308
Is Outsourcing Right for You? .....309
Do your goals and timeline indicate a clear need to outsource? .....310
Do you need to run or own the system? .....312
Analyzing your resources .....314
Money, money, money: Comparing outsourcing and internal costs .....316
Performance anxiety: Can you build a network that works? .....317
Finding the Perfect Match .....318
Figuring out the RFP process .....318
Spelling out your needs in an RFP .....320
Selecting potential outsourcing partners .....326
Evaluating responses to your RFP .....327
Sealing the Deal with an SLA .....327
Drafting the initial SLA .....328
Negotiating an SLA with a vendor .....331

***Part VI: The Part of Tens .....333***

**Chapter 18: Ten (Or So) Equipment Vendors .....335**

Alien Technology .....335
ACCU-SORT .....336
Applied Wireless Identifications (AWID) .....336
FOX IV Technologies .....337
Impinj .....337
Intermec Technologies .....338
MARKEM .....339
Symbol Technologies, Inc. (Formerly Matrics) .....339
ODIN technologies .....340
OMRON electronics .....340
SAMSys Technologies .....341
Texas Instruments (TI) .....341
ThingMagic .....342

**Chapter 19: Ten Web Sites for Information on RFID** ..... 343

RFID Journal Online ..... 344
EPCglobal ..... 344
IDTechEx ..... 345
RFID Solutions Online ..... 345
RFID Exchange ..... 345
RFID Update ..... 346
Auto-ID Labs ..... 346
Auto-ID Lab @ Adelaide ..... 346
The RFID Gazette ..... 347
UCLA's RFID@WINMEC site ..... 347
Slashdot ..... 347

**Chapter 20: Ten Tips from the Experts** ..... 349

Chris Fennig, ODIN technologies ..... 349
Joe White, Symbol Technologies (Formerly Matrics, Inc.) ..... 350
Duncan McCollum, Computer Sciences Corporation (CSC) ..... 351
Dr. Daniel Engels, MIT Auto-ID Labs ..... 352
Dr. Patrick King, Michelin Tire Corporation ..... 353
Steve Kowalke, ACCU-SORT Systems ..... 353
Team Tag-IT, Texas Instruments ..... 354
Kevin MacDonald, Lead RFID Architect, Sun Microsystems ..... 354
Mark Nelson, Savi Technology ..... 355

**Chapter 21: Ten (Or So) RFID Standards and Protocols** ..... 357

EAN.UCC ..... 357
EPCglobal ..... 358
UCCnet ..... 358
ISO/IEC JT1/SC17 ..... 359
ISO/IEC JTC1/SC31/WG4 ..... 360
AIAG ..... 361
Container Shipments ..... 361
Container Security Initiative (CSI) ..... 361
Smart and Secure Tradelanes ..... 362

***Appendix: Glossary of Electrical, Magnetic, and Other Scientific Terms*** ..... *363*

***Index*** ..... *373*

# Introduction

Somewhere, separated from you by just a few degrees, is not Kevin Bacon, but an 800-pound gorilla demanding that you adopt radio frequency identification, or RFID — a technology you may have never even heard of until just a few months ago. Chances are that gorilla wears a stylish blue smock with a yellow smiley face on it and greets you with a "Welcome to Wal-Mart." If not Wal-Mart, the US Department of Defense, Target, Albertsons, Best Buy, Tesco, Metro, the FDA or a number of other companies may be requiring you to implement this technology by a certain deadline. If you don't have a mandated deadline for adopting RFID, consider yourself lucky. You can discover and make decisions about this exciting technology based on your normal process for evaluating new business tools.

Whatever your situation is, you either want or need to set up an RFID network. So you went out and picked up *RFID For Dummies* and are ready to go — yippee!

## About This Book

This is a book that is on a mission to take the confusion out of RFID. RFID is based on well-known laws of physics. It's easy to understand how things work after you get your arms around those basics. The better news is that the technology works really well if you know what you're doing. So without sending you to MIT for a couple of years of RF engineering school, this book explains everything you need to know to start setting up and deploying your own RFID network — what more could you ask for?

## Who This Book Is For

Whether you are just curious, scared, worried, or simply mad at the prospect of implementing yet another new technology — even if you know nothing about RFID — *RFID For Dummies* is here to help. And, unlike a similar promise by the IRS, this book really will help. You find out what RFID is, what it does, and how it works. I guide you through the concepts and ideas in plain English, walk you through the basics of RFID from a business perspective, and speculate on where this technology is headed (although I do, from time to time,