RFID Bar Code Interoperability

GS1 Guideline

Issue #1, 10 August 2012

Document Summary

Document Item	Current Value
Document Title	RFID Bar Code Interoperability GS1 Guideline
Date Last Modified	10 August 2012
Current Document Issue	Issue #1
Status	Ratified
Document Description	

Contributors

Name	Organization
Cynthia Poetker, working group co-chair	Abbott Laboratories Inc.
Dirk Rodgers, working group co-chair	Cardinal Health
Michael Sarachman, working group facilitator	GS1 Global Office
Ken Traub, document editor	Ken Traub Consulting LLC
Sprague Ackley	Intermec Technologies Corp.
Dipan Anarkat	GS1 Global Office
Shirley Arsenault	GS1 Global Office
Henri Barthel	GS1 Global Office
Bob Bersani	GS1 Global Office
Chuck Biss	GS1 Global Office
Ron Bone	McKesson
Daniel Carmona	GS1 Mexico
Robert Celeste	GS1 US
James Chronowski	GS1 US
Kevin Dean	GS1 Canada
Ray Delnicki	GS1 US
Daniel Eumaña	GS1 Mexico
Dawn Fowler	Edwards Lifesciences
Mark Frey	GS1 Global Office
Andreas Fuessler	GS1 Germany
Heinz Graf	GS1 Switzerland
Scott Gray	GS1 Global Office
Michaela Haehn	GS1 Germany
Andrew Hearn	GS1 Global Office
John Howells	HDMA
Yoshihiko Iwasaki	GS1 Japan

Name	Organization
Janice Kite	GS1 Global Office
Sean Lockhead	GS1 Global Office
Kevan MacKenzie	McKesson
Reiko Moritani	GS1 Japan
Alice Mukaru	GS1 Sweden
Andrew Osborne	GS1 UK
Benjamin Östman	GS1 Finland
Craig Alan Repec	GS1 Global Office
John Roberts	GS1 US
Sue Schmid	GS1 Australia
Eugen Sehorz	GS1 Austria
Frank Sharkey	GS1 Global Office
Yuko Shimizu	GS1 Japan
Jack Sparn	iGPS Company LLC
Joe Spreitzer	Target Corporation
Sylvia Stein	GS1 Netherlands
Sue Thompson	National Council for Prescription Drug Programs
Lionel Willig	GS1 France
Roman Winter	GS1 Germany
Ruoyun Yan	GS1 China
Mike Zupec	Abbott Laboratories Inc.

Log of Changes in Issue #1

Issue No.	Date of Change	Changed By	Summary of Change
1	10 August 2012		First issue ratified.

Disclaimer

Whilst every effort has been made to ensure that the guidelines to use the GS1 standards contained in the document are correct, GS1 and any other party involved in the creation of the document HEREBY STATE that the document is provided without warranty, either expressed or implied, of accuracy or fitness for purpose, AND HEREBY DISCLAIM any liability, direct or indirect, for damages or loss relating to the use of the document. The document may be modified, subject to developments in technology, changes to the standards, or new legal requirements. Several products and company names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

Table of Contents

List of Figures

List of Tables

1. Introduction

The GS1 System provides globally unique identification keys to identify and describe a variety of business objects, including products and services, logistic units, locations, assets, documents, and service relationships. GS1 Identification Keys are designed to be used in any business application, database, or business message that needs to refer to a real-world business object in an unambiguous way. GS1 Identification Keys are globally unique identifiers that name business objects, and when combined with descriptive data facilitate a common understanding of these objects between trading partners.

The GS1 System also provides a number of different data carriers that provide a means of physically affixing GS1 Identification Keys and other data to a physical object so that the data may be captured without the need for manual data entry. GS1 data carriers include a variety of 1-D and 2-D bar codes, as well as various types of Radio-Frequency Identification (RFID) tags.

An end user or system designer is often confronted with a picture similar to the figure below:

Figure 1. Integrating Data Carriers to Business Applications

The left hand side of the figure shows various data carriers used to affix GS1 Identification Keys and other data to physical objects. The right hand side of the figure shows various business processes and systems that will use data captured from those data carriers.

2. Scope

This guideline addresses how end users and solution providers can design systems that confront the challenge illustrated in Figure 1 to achieve the greatest degree of interoperability possible. This guideline describes three best practices for doing so:

1. Design business-level applications, databases, and messages to be independent of the data capture method and the data carriers used.
2. Confine the use of data carrier-specific representations to the lowest levels of implementation architecture.
3. Adopt best practices for implementing translations between data carrier-specific representations and application-level representations.

These three practices are expanded upon in the following three sections, after which specific scenarios are illustrated.

This GS1 Guideline provides information to end users and solution providers that use GS1 data carriers including bar codes and radio frequency identification (RFID) tags, specifically, data carriers that include serialized data. The guideline provides recommendations for best practices designed to lead to the highest degree of interoperability when using these data carriers, especially in any of the following situations:

- When both bar codes and RFID tags are used together to label physical objects
- When it is necessary to transfer information from a bar code to an RFID tag or vice versa
- When deploying information systems that may read and write bar codes and RFID tags

The best practice recommendations in this guideline are derived relevant GS1 Standards including the GS1 General Specifications and the GS1 EPC Tag Data Standard in alignment with the GS1 System Architecture. This guideline does not attempt to reproduce all of the relevant information found in those documents, nor serve as a replacement for them. Where appropriate, this document refers the reader to the relevant standards to provide complete detail on the topics discussed here. Users of this guideline should be familiar with the GS1 General Specifications and GS1 EPC Tag Data Standard.

3. Data Carrier Independence

A single GS1 Identification Key such as a Serial Shipping Container Code (SSCC) may exist in many different concrete representations, some specific to a data carrier, some independent of data carrier.

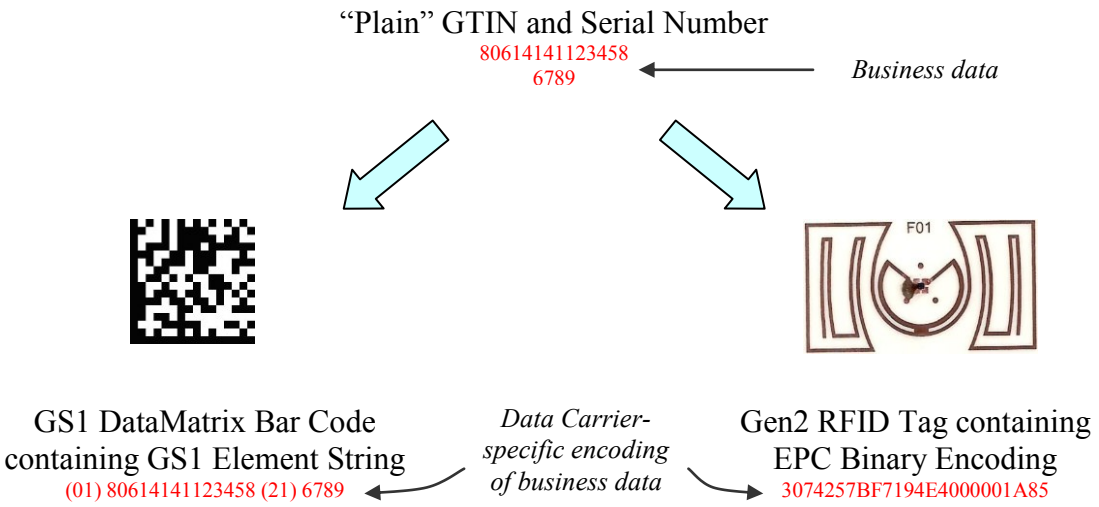
In the GS1 System, a very clear distinction is made between *data* and a *data carrier*. GS1 Standards provide normative definitions of various data elements and these definitions hold true regardless of the form that data takes or how data is transmitted. A “data carrier” as that term is used in GS1 Standards is a means of physically affixing data to a physical object so that the data can be captured without human data entry. The industry term “Automatic Identification and Data Capture (AIDC)” is used to refer to data carriers and the associated technology and business processes for using them. In the GS1 System, data carriers include a variety of types of bar codes and radio frequency identification (RFID) tags.

A principle of GS1 standards is that data elements are defined in a data carrier neutral way so that their semantics is the same regardless of what data carrier is used to affix them to a physical object (and also the same outside of a physical data carrier, such as in an electronic message). However, there is a multitude of different data carriers, each optimized for a particular set of physical and performance constraints arising in the real world. Standards for each data carrier therefore define a carrier-specific representation of carrier-neutral data elements, allowing those data to be encoded in a manner compatible with the physical constraints of the carrier.

For example, in a GS1 DataMatrix bar code data elements are first arranged into a “concatenated element string” that frames each data element with a short identifying character sequence and (where necessary) separator characters, then at a lower level encodes the resulting sequence into a pattern of dark and light squares. In a GS1 Gen 2 RFID tag, those same data elements are separated into different memory banks, then at a lower level encoded into a highly compacted binary form and stored into an electronic memory circuit. In both bar code and RFID, the lower level encoding also includes

an indicator that the data carrier contains GS1 System data, to distinguish it from non-GS1 uses of the same data carrier technology. This illustrated in Figure 2, below.

Figure 2. GTIN and Serial Number as Encoded Into Two Different Data Carriers



Likewise, data may be used in a variety of business application contexts, and again the data elements are defined so that their meaning is the same regardless of how they are used at a business level. The way that data elements are represented internally within different business applications may differ depending on the context and the technology involved. For example, in a database that is concerned only with product identification, there may be a data column that contains Global Trade Item Number (GTIN) to identify a product, represented as a 14-digit string. In an Electronic Product Code Information Services (EPCIS) database that tracks a variety of physical objects moving through a supply chain, that same GTIN is combined with a serial number and represented as a Uniform Resource Identifier (URI) that allows both GTINs and other types of identifiers to co-exist within the same data.

Different forms are discussed and illustrated in the next sections.

3.1. Application-Level Syntaxes

GS1 Standards define three different data syntaxes for representing GS1 Identification Keys that are recommended for use at an application level. "Syntax" refers to a particular way of writing the value of a GS1 Identification Key as a sequence of characters. Application-level syntaxes have the following three properties:

- Accommodates every possible value of a GS1 Identification Key without limitation, and so it is capable of representing a key read from any data carrier.
- Does not include additional information that is specific to a particular type of data carrier.
- Provides only one possible way to represent each distinct key value within the syntax. This means that an application can determine whether two values refer to the same real-world entity by a simple string comparison, with no additional normalization or parsing required.

These properties make these syntaxes suitable for use in business applications, databases, and messages to ensure the greatest degree of interoperability. Applications may use these syntaxes regardless of what data carriers are used.

Three application-level syntaxes are described in Section 4.4 of the GS1 System Architecture and are summarized in Table 1, below. Examples are provided later in this section.

Table 1. Application-level Syntaxes

Syntax	Description	Where Used (at business application level)
"Plain"	Digits or characters comprising the key value, with no punctuation, headers, or other characters	GS1 eCOM (EANCOM and XML) GS1 GDSN messages User-defined databases and messages where only one type of key is allowed
GS1 Element String	A short (2-4 character) "Application Identifier" that indicates the type of GS1 Identification Key, followed by the digits or characters of the key value itself. In the case of GTIN+serial or GLN+extension, the string contains two such sequences, concatenated.[1]	User-defined messages where any GS1 Identification Key is allowed, or where multiple data elements need to be conveyed as a single unit (e.g., GTIN and serial number)
Pure Identity Electronic Product Code Uniform Resource Identifier (EPC URI)	An Internet Uniform Resource Identifier (URI) beginning with `urn:epc:id:…` and the remainder having a syntax defined by the GS1 EPC Tag Data Standard	GS1 EPCIS messages and databases User-defined databases and messages where any type of identifier for a specific physical object is permitted

Note: in this guideline, the only "plain" syntaxes or GS1 Element Strings under consideration are those that have an equivalent representation as EPCs, as defined in Section 6 of the GS1 EPC Tag Data Standard. These are serialized keys with no additional attributes (e.g., GTIN + serial (SGTIN), SSCC, GRAI with serial, etc.).

Table 2 shows a GTIN and serial number represented in each of the three application-level syntaxes.

Table 2. Examples of GTIN and Serial Number (SGTIN) in Different Syntaxes

Representation	Example	Comment
"Plain"	80614141123458 6789	The GTIN and serial number are here shown separately
GS1 Element String	0180614141123458216789	The Application Identifier "01" precedes the 14-digit GTIN, and "21" precedes the serial number

[1] The GS1 General Specifications refer to the combination of a single Application Identifier and its corresponding data value as an "element string," and the concatenation of two or more such element strings as a "concatenated element string." As this guideline only considers element strings that have an equivalent representation as EPCs as defined in Section 6 of the GS1 EPC Tag Data Standard, this will either be a single element string, or a concatenated element string with two Application Identifiers in the case of GTIN+serial or GLN+extension. For the sake of brevity, this guideline uses the term "GS1 Element String" to refer to either of these possibilities.

Representation	Example	Comment
Pure Identity EPC URI	urn:epc:id:sgtin:0614141.812345.6789	In the Pure Identity EPC URI, the digits of the GTIN comprising the GS1 Company Prefix are separated from the others, the indicator digit or leading zero is moved to immediately precede the Item Reference, and the check digit is omitted. See Section 4.1.

Table 3, below, shows an SSCC represented in each of the three application-level syntaxes.

Table 3. Examples of SSCC in Different Syntaxes

Representation	Example	Comment
“Plain”	806141411234567896	
GS1 Element String	00806141411234567896	The Application Identifier “00” precedes the 18-digit SSCC
Pure Identity EPC URI	urn:epc:id:sscc:0614141.8123456789	In the Pure Identity EPC URI, the digits of the SSCC comprising the GS1 Company Prefix are separated from the others, the extension digit is moved to immediately precede the Serial Reference, and the check digit is omitted. See Section 4.1.

Table 4, below, shows a GRAI, including serial number, represented in each of the three application-level syntaxes.

Table 4. Examples of GRAI in Different Syntaxes

Representation	Example	Comment
“Plain”	06141411234526789	The first 13 digits of the GRAI identify the type of returnable asset, and the remaining digits (“6789” in this example) are the serial number
GS1 Element String	8003006141411234526789	The Application Identifier “8003” and a filler “0” digit precedes the GRAI
Pure Identity EPC URI	urn:epc:id:grai:0614141.12345.6789	In the Pure Identity EPC URI, the digits of the GRAI comprising the GS1 Company Prefix are separated from the others, the filler “0” and check digit are omitted, and the serial number is separated. See Section 4.1.

Examples of other types of GS1 Identification Keys may be found in Appendix E.3 of the GS1 EPC Tag Data Standard.

The reason that three different syntaxes are defined is that each is designed to accommodate a different range of key types within a single context. That is, if a given database column or message field is defined to hold values of a given syntax, then the choice of syntax determines what kind of keys may appear in that database column or message field. Table 5 summarizes this.

Table 5. Data Content of Syntaxes for GS1 Identification Keys

Syntax	What It Can Hold
"Plain"	Any value of a particular GS1 Identification Key. As the syntax itself does not indicate the type of key, the context in which this syntax is used must specify what type of key is expected. For example, an XML message may have an element `<gtin>`, and that element is expected to hold a GTIN in "plain" syntax. The data dictionary for this XML message must specify that the contents is a GTIN.
GS1 Element String	Any GS1 Identification Key, specifically any Class 1 or Class 2 key as those terms are defined in the GS1 System Architecture, Section 4.3. The Application Identifier included in this syntax identifies what type of key is used in any particular instance; the context in which this syntax is used establishes that the syntax is a GS1 Element String and therefore GS1 Identification Keys are the only types of keys that are allowed.
Pure Identity Electronic Product Code Uniform Resource Identifier (EPC URI)	Any identifier for which an EPC URI representation is defined in the GS1 EPC Tag Data Standard. This includes those GS1 Identification Keys (Class 1 and Class 2) that correspond to specific physical objects as well as certain other identifiers from non-GS1 identification systems (Class 3). The context in which this syntax is used typically allows *any* URI to be used, including EPC URIs as well as other types of URIs defined entirely outside GS1 Standards (Class 4).

Here are examples of how the different application-level syntaxes are used in GS1 Standards:

- In eCom XML, a GTIN identifying a product in a purchase order is expressed like this:

  `<globalTradeItemNumber>`**`80614141123458`**`</globalTradeItemNumber>`

  The contents of the `<globalTradeItemNumber>` (in bold red) is a GTIN in "plain" syntax.

- In EANCOM, a GTIN identifying a product in a purchase order is expressed like this:

  `LIN+1++`**`4000862141404`**`:SRV'`

  The contents of the `LIN` segment (in bold red) is a GTIN in "plain" syntax.

- In EPCIS, a Serialized GTIN identifying a specific instance of a trade item in an Object Event is expressed like this:

  `<epc>`**`urn:epc:id:sgtin:0614141.812345.400`**`</epc>`

  The contents of the `<epc>` element (in bold red) is an SGTIN in EPC URI syntax.

3.1.1. Note about GTIN Variations

Throughout this document the GTIN is illustrated as containing 14 digits. In fact, the GTIN can be either 8, 12, 13, or 14 digits in length. As specified in Section 2.1.2.1.1 of the GS1 General Specifications a GTIN that is shorter than 14 digits can be represented as a 14-digit number by prepending as many zero digits as necessary to make 14 digits. Unless otherwise noted, where a 14-digit GTIN is illustrated in this document, it should be understood to stand for any GTIN, padded to 14 digits as necessary.

There is also a bar code symbol called the UPC-E which contains 8 digits. These eight digits are not a GTIN-8, but rather a compressed form of a GTIN-12 that meets certain criteria. Bar code readers typically expand a UPC-E into 12 digits before reporting to an application and then this GTIN-12 can be treated as above. See Section 7.11 of the GS1 General Specifications.

3.2. Data Carrier Specific Forms

The previous section illustrated the representations of GS1 Identification Keys that are recommended for use at the application level. As noted earlier, for each data carrier there are also carrier-specific representations which allow the keys to be encoded in a manner compatible with the physical constraints of the carrier. For greatest interoperability, these carrier-specific representations must not be used at the business level, but instead confined to the lower levels of a data capture implementation architecture as discussed in Section 4. The best practices for translating between the carrier-specific and application-level representations are given in Section 4.1.

Table 6 summarizes the data carrier-specific forms and describes how each fails to achieve one or more of the three properties from Section 3.1 that distinguish the application-level forms.

Table 6. Data Carrier-Specific Forms

Carrier-specific Form	Description	Carrier-specific Properties
Bar code scanner output	Many bar code scanners can be configured to output a string consisting of a 3-character "symbology identifier" code that identifies what type of bar code (i.e., symbology) is read and whether it represents GS1 System data, followed by the characters contained in the bar code. These characters include Application Identifiers, data values, and possibly separator characters. (Note that some bar code scanners can also be configured to output "plain" syntax as defined in Section 3.1, without symbology identifiers and in some cases without Application Identifiers; these comments do not apply in that case. GS1 recommends that Symbology Identifiers be enabled and transmitted with every bar code symbol read.)	The symbology identifier is extra information beyond the GS1 Identification Key, and is carrier-specific The same GS1 Identification Key will have a slightly different output string depending on what kind of bar code is read, so output strings cannot be directly compared.

Carrier-specific Form	Description	Carrier-specific Properties
EPC Tag URI	The EPC Tag URI looks similar to a Pure Identity EPC URI, but includes additional information that is carried within the EPC memory bank of an RFID tag. This additional information includes the “filter” value that helps RFID readers deal more efficiently with large tag populations, selection of alternative encoding sizes, etc. The EPC Tag URI is used in low-level software that needs to know the complete contents of the EPC memory bank, including RFID-specific control information. See Section 12 of the GS1 EPC Tag Data Standard.	The “filter” value and other RFID controls are information beyond the GS1 Identification Key and is carrier-specific Depending on what encoding size is chosen, there may be restrictions on the GS1 Identification Key value (e.g., the serial number is restricted in the SGTIN-96 encoding) The same GS1 Identification Key may have different EPC Tag URIs depending on the filter value, encoding size, etc, so EPC tag URIs cannot be directly compared
EPC Binary Encoding	The EPC Binary Encoding shows the encoded contents of the EPC memory bank of an RFID tag, bit-for-bit. It is often written, equivalently, in hexadecimal rather than binary. In comparison to the EPC Tag URI, it is much harder for humans to understand	The binary data is highly specific to the RFID data carrier, which uses electronic memory components to hold data The binary encoding includes the “filter” value and other RFID controls that are information beyond the GS1 Identification Key and so is carrier-specific Depending on what encoding size is chosen, there may be restrictions on the GS1 Identification Key value (e.g., the serial number is restricted in the SGTIN-96 encoding) The same GS1 Identification Key may have different EPC Tag URIs depending on the filter value, encoding size, etc, so binary encodings cannot be directly compared
RFID User Memory Encoding (“packed objects”)	The RFID User Memory Encoding is a bit-level representation of data used in the User memory bank of an RFID tag. It is capable of holding all of the same data elements that can be contained in a GS1 bar code in a very highly compressed form	The binary data is highly specific to the RFID data carrier, which uses electronic memory components to hold data The user memory encoding includes a variety of framing data used to guide decompression of the data and other RFID-specific features and so is highly data carrier specific. The compression algorithms allow for a variety of ways to encode the same data value, so user memory contents cannot be directly compared

Figure 4 in Section 5 shows examples of each of the data carrier-specific forms for a GTIN and Serial Number (SGTIN).

3.3. Considerations in Assigning Serial Numbers for GTINs

Certain interoperability concerns arise in connection with assigning serial numbers for GTINs. These issues affect any party responsible for assigning serial numbers for GTINs, typically the brand owner of a product or a third party to whom the brand owner delegates the process of assigning serial numbers. Considerations arise because of the variable-length, alphanumeric structure of the GTIN serial number, and because of limitations of 96-bit RFID tags.

The AI (21) serial number associated with a GTIN is defined in the GS1 General Specifications as an alphanumeric string of between 1 and 20 characters in length. Because it is an alphanumeric string, the character zero ('0') is treated no differently than any other character. This means that serial numbers "7", "07", and "007" are all considered to be *different* serial numbers, just as serial numbers "7", "17", and "117" are all different, and "7", "A7", and "AA7" are all different.

This fact sometimes causes problems if systems inadvertently treat the serial number as a decimal numeral, and delete leading zeros. For example, if the serial numbers "7", "07", and "007" are read into a typical spreadsheet program, the spreadsheet is likely to interpret them as numbers and display all of them as "7". If the spreadsheet is saved in that form, the serial numbers "07" and "007" will have inadvertently been *changed* to a different number. Whenever serial numbers are stored or manipulated in software, it is essential that they be handled as character strings rather than as numbers, to avoid inadvertent data corruption of this kind. One way to defend against the possibility of such errors is to avoid issuing serial numbers that begin with "0" characters.

Another consideration comes into play when serial numbers are encoded into bar codes. Because the serial number is of variable length, the number of characters in the serial number can affect the printed size of the bar code. Manufacturing production processes and symbol specifications often impose constraints on the minimum and/or maximum size of a printed bar code, both to preserve the design of the overall package artwork and to reduce the complexity of quality assurance of on the printed bar code symbol. This results in constraints on the range of serial numbers that my be used. For example, if a particular process requires a fixed size symbol, manufacturers may wish to adopt a serial number assignment policy that only uses serial numbers of a fixed length; e.g., always using 11 characters. When serial numbers are assigned using only digits, manufacturers may wish to combine this policy with a policy that avoids leading zeros as discussed above, so a manufacturer may limit its serial numbers to the range 10000000000 – 99999999999 (to use an 11-digit example).

Yet another consideration comes into play when 96-bit RFID tags are used. It is not possible to accommodate every possible value of a 1–20 character alphanumeric serial number in 96 bits, and so the SGTIN encoding for 96-bit tags specified in the GS1 EPC Tag Data Standard imposes a limitation on what serial numbers can be used in a 96-bit tag. Specifically, to be encoded in a 96-bit RFID tag a GTIN serial number must be all numeric (i.e., the only characters permitted are the digits 0 through 9), the first digit must not be a "0", and the value when read as a decimal numeral must be less than or equal to 274877906943 (equal to $2^{38}-1$). For example, the serial numbers "7", "1234", and "274877906943" may all be encoded in a 96-bit tag, but the following serial numbers cannot: "07" (starts with a leading zero), "274877906944" (too large in magnitude), "A123" (contains a character other than a digit).

When GTIN serial numbers need to be encoded in both bar codes and 96-bit RFID tags, a manufacturer may wish to adopt a policy that combines the above considerations. For example, assigning numeric serial numbers in the range 10000000000 – 99999999999 results in both a fixed length (11 digits) that keeps the bar code symbol size constant and also fits within the constraints of a 96-bit RFID tag (numeric, no leading zeros, and within the magnitude limit). If a 12-digit serial number is preferred, the range 100000000000 – 274877906943 could be used. A 13-digit or longer serial number could not be used because no 13-digit number fits within the constraints of the 96-bit RFID tag.

Note that RFID tags that have a higher memory capacity can accommodate the 198-bit encoding of SGTIN defined in the GS1 EPC Tag Data Standard. This encoding does not have any restrictions as to the serial number: any 1–20 character alphanumeric serial number as defined in the GS1 General Specifications may be encoded into a 198-bit (or larger) tag.

Regardless of what policy is adopted for the *assignment* of serial numbers, for full interoperability it is essential that all information systems be capable of accepting the full 1–20 character alphanumeric serial number, and treat it as a text field so that leading zeros are never removed.

3.4. Summary – Interoperability Principles

- Business applications, messages, and databases should be designed to accept data from any data carrier; in particular, this means that applications and databases should be designed to accept the full range of data values defined by GS1 Standards. The restrictions on data values that certain data carriers impose should not be carried through to this level.

  Rationale: Accepting the full range of data values ensures that the application or database can accept data no matter what data carrier is used.

  Example (good practice): A database includes a column for serial number (associated with a GTIN) that is defined to accept between 1 and 20 alphanumeric characters. This database will be capable of holding any serial number, regardless of data carrier.

  Example (bad practice): A database includes a column for serial number that is defined to accept up to 12 numeric digits. This database can hold any serial number read from a 96-bit RFID tag, but not from other types of RFID tags or from bar codes because those carriers may contain a serial number that contains alphanumeric characters and/or is longer than 12 characters.

- Ideally, business applications, messages, and databases should only use the application-level syntaxes defined in Section 3.1.

  Rationale: Using an application-level representation avoids building in a dependence on a specific data carrier or inadvertently relying upon carrier-specific control information.

  Example (good practice): A business message intended to convey a serialized GTIN uses the EPC Uniform Reference Identifier (URI) representation: `urn:epc:id:sgtin:0614141.112345.400`. This representation can be derived from either a bar code or an RFID tag, and does not indicate what data carrier was used.

  Example (bad practice): A business message intended to convey a serialized GTIN uses the EPC binary encoding for RFID tags: `3034257BF46DB64000000190` (hexadecimal). This representation is specific to 96-bit RFID tags, and includes information that is RFID-specific (the "filter" value used by readers to read tags more efficiently). Using this representation in a business message will be confusing when read by humans and make it difficult to interoperate with bar codes.

- Designers of business applications, messages, and databases should consider carefully which of the three application-level syntaxes are most appropriate to use. Some considerations include:

  - In general, a more restrictive syntax is appropriate when the nature of the application is such that only one key (e.g., SSCC but not GRAI, GIAI, etc) could possibly make sense in that application; e.g., "plain" syntax is a good fit for a point-of-sale application if that application only scans GTINs. A less restrictive syntax is appropriate when an application can deal with different identification types generically; e.g., the Pure Identity EPC URI is a good fit for a visibility application that tracks the movements of a variety of types of objects (including trade items, assets, documents, etc) within an airport.

  - Using a more restrictive syntax (e.g., "plain" syntax instead of GS1 Element String) may simplify application processing, and provides a degree of built-in validation that the correct

type of key is used. For example, if a business message contains a GTIN field it is clear that an SSCC is not appropriate in that context.

- Using a less restrictive syntax (e.g., EPC URI or GS1 Element String instead of "plain" syntax) may provide a degree of "future-proofing" in case application requirements evolve to include new types of identification that were not considered when the application was first designed.
- To the degree that a new application is expected to interoperate with other applications whose preferred syntax is already defined, aligning with that preference can reduce the amount of syntax translation required. For example, an application that expects to send and receive EPCIS messages might choose to use Pure Identity EPC URIs internally, since Pure Identity EPC URI syntax is used in EPCIS messages.

- Applications must not add or remove leading zeros to serial numbers.

  Rationale: Leading zeros in serial numbers are significant characters, no different from any other character. "7", "07", and "007" are all different serial numbers. Adding or removing leading zeros changes the serial number.

  Example (good practice): An information system assigns serial number 6789 to an instance of a product. This is encoded into a 96-bit RFID tag, and also encoded into a bar code using AI (21) 6789.

  Example (bad practice – not standards compliant): An information system assigns serial number 006789 to an instance of a product. The serial number 6789 is encoded into a 96-bit RFID tag (because 006789 is not encodable). AI (21)00006789 is encoded into a bar code because it is desired to have an 8-digit serial number for symbol size reasons. This is not correct, because 006789, 6789, and 00006789 are all *different* serial numbers according to the GS1 General Specifications. If the application requirement is for an 8-digit serial number that can also be encoded into a 96-bit RFID tag, an 8-digit number beginning with a non-zero digit should be used; e.g., serial number 10006789.

- While the standards support serial numbers beginning with "0", applications that assign serial numbers for use with GTIN should avoid serial numbers that begin with a "0" character. If 96-bit RFID tags are to be used, serial numbers should be assigned that fit within the encoding constraints of the 96-bit SGTIN format as defined by the GS1 EPC Tag Data standard.

  Rationale: leading zeros may be subject to errors in system implementation that wrongly delete leading zeros. Errors of this kind cannot occur if serial numbers do not have leading zeros in the first place. Fitting within the encoding constraints of 96-bit RFID tags is necessary if 96-bit RFID tags are used.

4. Architecture

Section 5 of the GS1 System Architecture (February 2012) discusses in detail implementation architecture considerations for data capture. When referring to the process of reading or writing bar codes or RFID tags, the term data capture is referred to as Automatic Identification and Data Capture (AIDC) in GS1 General Specifications. "Data capture" as used in this guideline is broader than AIDC, as it refers to the entire business process of capturing data while handling physical objects, including interaction with AIDC data carriers as well as other interactions with humans, information systems, etc. Where the term "data capture" in this guideline is associated with bar codes or RFID tags, this corresponds to AIDC as that term is used in the GS1 General Specifications.

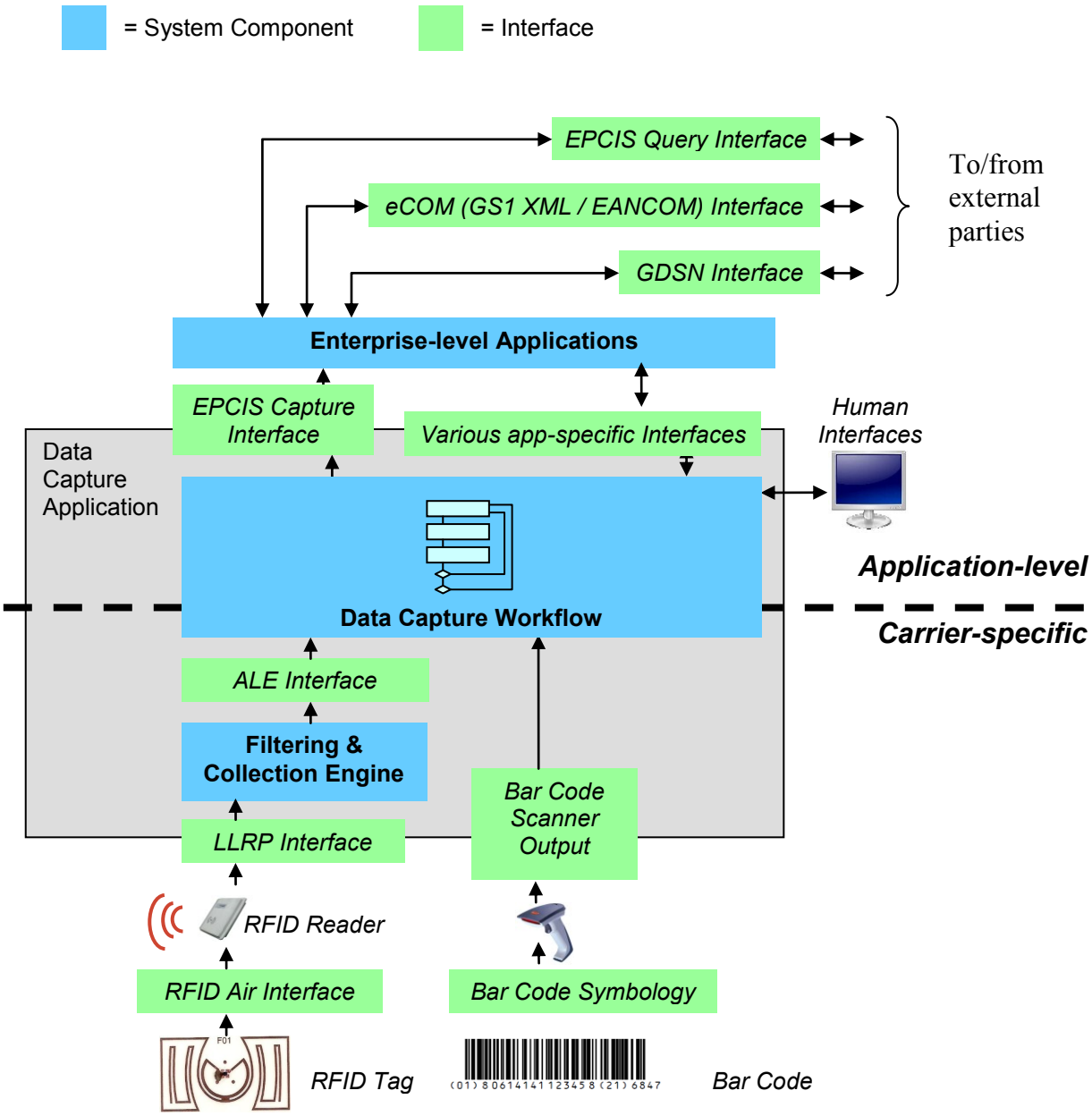
A recurring architectural pattern is that physical data capture takes place within the context of a data capture workflow, which is a localized business process involving the handling of physical goods. Examples of data capture workflows include: checking out trade items at point-of-sale, receiving logistic units into inventory, assigning unique serial numbers and marking them on products or packaging (here the data captured is the association of the serialized identifier to the physical object),

inventory counting of fixed assets, and so forth. Data capture workflows mostly take place at the “edge” of a business – in physical plant such as factories, warehouses, retail stores, as opposed to business workflows that exist virtually within applications running in data centres.

Within a given data capture workflow there may be many individual interactions with physical data carriers. A data capture workflow may also involve interaction with humans, as well as “back end” information systems such as Enterprise Resource Planning (ERP), Warehouse Management System (WMS), etc. All these things combine to create a business context within which the data capture workflow takes place and this gives meaning to the act of capturing the data from the data carrier.

As described in Section 5.1 of the GS1 System Architecture, this leads to a typical architecture for a data capture application illustrated in Figure 3, below.

Figure 3. Typical Data Capture Architecture



Data including GS1 Identification Keys exists at all levels of the picture above, both within the various system components (the blue boxes) and across the various interfaces (the green boxes). In designing or configuring any of these components an implementer may be faced with a choice of data representation.

The most important best practice for interoperability is to confine data carrier-specific representations to the lowest levels of this architecture, so that a change in data carrier has the least impact on the overall architecture. The dashed line in the diagram shows where data carrier-specific and application-level representations should generally be used.

The dashed line is shown passing through the middle of the Data Capture Workflow layer, suggesting that a data capture workflow may need to work with both carrier-specific and application-level data

representations. For example, a workflow that is reading RFID tags affixed to cases of product aggregated to a pallet may use the RFID "filter" value to help infer which data value was read from the pallet tag versus from the cases. The "filter" value is RFID-specific information that the data capture workflow needs to do its job. By the time data goes up to an enterprise application, however, the filter value should not be included. Instead, the enterprise-facing data should distinguish case and pallet identifiers in a carrier-independent way; for example, by defining a business message that has a separate data field for the pallet identifier as compared to the case identifiers. (The EPCIS Aggregation Event message, for example, has such a structure.)

To illustrate the interoperability benefits, consider what happens in the above example if the process changes to use bar codes instead of RFID tags. The workflow can no longer use filter values to distinguish pallet from case, as they do not exist in bar codes. Instead, the workflow necessarily will use a different method; e.g., instructing an operator to scan the pallet bar code before scanning the case bar codes. But when the data is reported to the enterprise application, the same business message that has one field for the pallet identifier and a different field for the case identifiers may be used; neither the message definition nor the consuming application need to be reworked as a consequence of changing the data carrier.

In many cases, the data capture workflow will be responsible for converting between data carrier-specific representations and application-level representations. In some cases, this conversion can be pushed to a lower level; for example, if the bar code or RFID reading device is capable of generating output in an application-level syntax and the data capture workflow does not need any carrier-specific information. The next section (Section 5) discusses best practices for carrying out such conversions when necessary.

4.1. Summary – Interoperability Principles

- Confine the use of data carrier-specific representations to the lowest possible level in the architecture.

  Rationale: The more confined the use of carrier-specific representations, the greater the part of the system that can interoperate with all data carriers. Less rework will be required if data carriers change.

  Example (good practice): A data capture workflow that uses RFID "filter" values to distinguish pallet-level and case-level tags sends data to a business application in a message format that has an explicit field for pallet identifier that is separate from case identifier, and those fields are filled in with application-level representations of the pallet and case identifiers. The filter value is not transmitted to the business application.

  Example (bad practice): A data capture workflow sends raw RFID tag read data in EPC binary format to a business application. The business application uses the "filter" value in the raw data to distinguish pallet from case. If the data carrier is changed from RFID to bar code, either the business application will require extensive rework, or the new bar code data capture logic will have to translate to an RFID-specific form using simulated RFID filter values that don't actually exist in the bar code.

5. Translation

As discussed in the previous sections, different representations of GS1 Identification Keys may exist in different parts of an implementation. In particular, data carrier specific forms are found within data capture hardware and software, and application-level forms are used in business applications, databases, and business messages. A good architecture limits data carrier dependences to the extent possible and ensures interoperability to the extent possible where it is not. In practice this means isolating the data carrier specific forms to the lowest level of the architecture possible. This implies that some translation is needed to transfer data between data carrier specific forms at the low level to

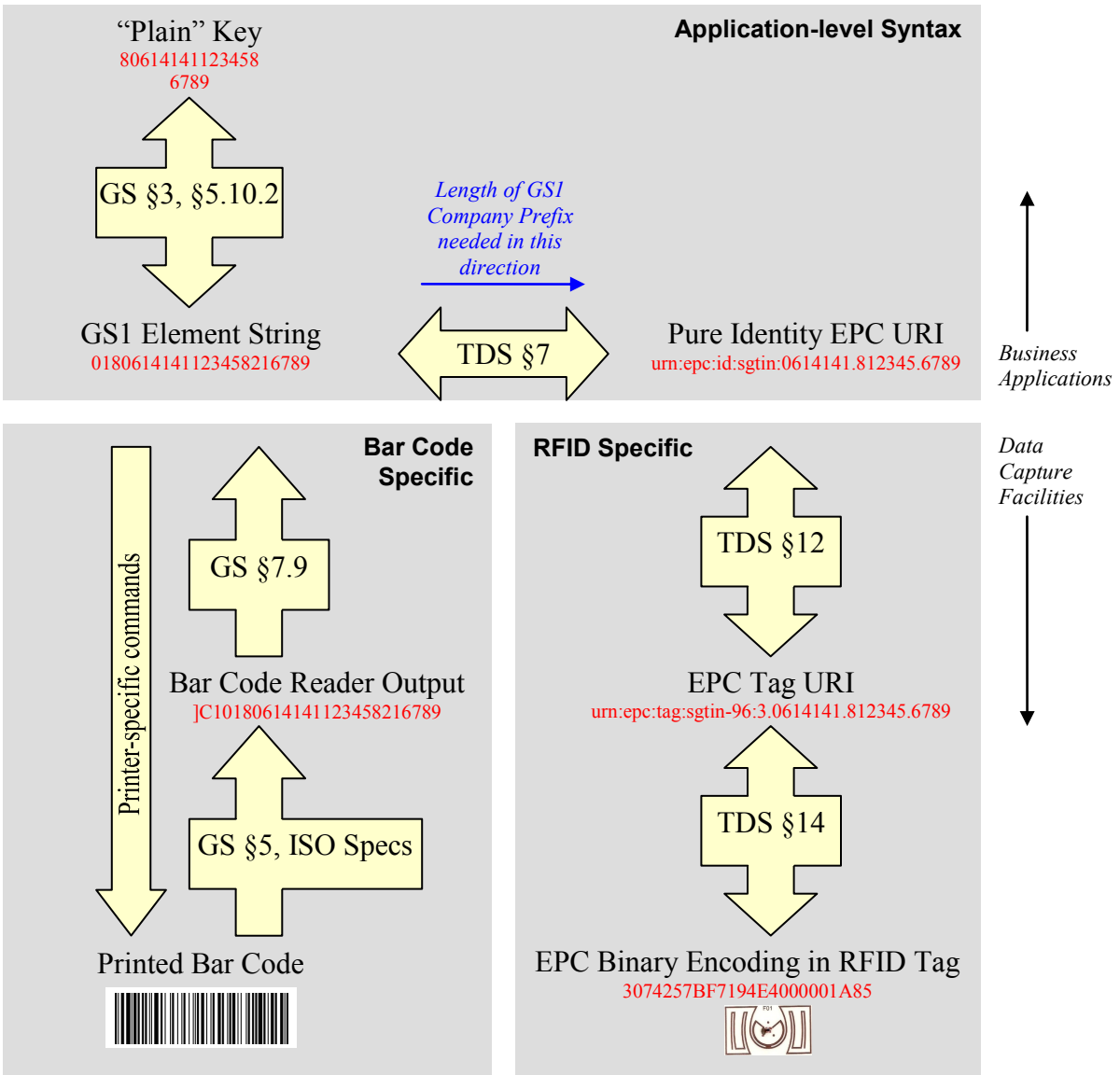
application-level forms at the high level, and vice versa. Translation may also be needed between different data carrier specific forms, as when bar code data is copied into an RFID tag or vice versa. Translation may also be needed between application-level forms, for example if data from an EPCIS message containing a Pure Identity EPC URI is to be copied into GS1 XML eCOM message requiring 'plain" syntax.

The details of the translations are specified in the GS1 EPC Tag Data Standard and the GS1 General Specifications. These translation procedures are typically implemented in software developed for this purpose or embedded within larger systems. In addition, the GS1 EPC Tag Data Translation Standard defines a machine-readable language that describes these translations. This translation language is used by some translation software "engines," enabling the translation rules to be easily updated as the standards evolve. Not all translation software uses the Tag Data Translation standard, however.

Figure 4 summarizes all of the various translations that may be needed and where they are defined in GS1 Standards.

Figure 4. Data Translations and Their Governing Standards

GS = GS1 General Specifications
TDS = GS1 EPC Tag Data Standard
§ indicates the section number where the translation is defined



When data is read from a bar code or RFID tag and presented to a business application, data flows from bottom to top in Figure 4. When a business application provides data so that a bar code can be printed or an RFID tag encoded, data flows from top to bottom. Each of the translations, therefore, is bidirectional, with one procedure to translate in the upward direction and a different procedure for the downward direction.

The following sections briefly describes the translations that occur along each arrow in Figure 4.

5.1. Translations from Bar Code to Business Application

The translations in this section are employed when a bar code is read to provide information to a business application and correspond to the upward path along the left side of Figure 4.

5.1.1. Bar Code to Bar Code Reader Output

A bar code is read by a scanning device where the light and dark graphical pattern is decoded to arrive at a sequence of characters that is output. This decoding procedure is typically embedded within the bar code reader hardware. The exact form of this output may vary according to how the device is configured. Most devices are capable of transmitting a string that begins with a 3-character "symbology identifier" that indicates what type of bar code was scanned. There may also be separator characters (ASCII GS characters) present in addition to the data. (See GS1 General Specifications, Section 5.) The example in Figure 4 shows the symbology identifier "]C1" (indicating a GS1-128 bar code was scanned and that data conforms to GS1 specifications), followed by the data payload.

5.1.2. Bar Code Reader Output to GS1 Element String

The output from the bar code scanner is translated into a GS1 Element String. For certain types of bar codes (e.g., GS1-128, GS1 DataBar, GS1 DataMatrix), this translation merely verifies the control character that indicates the presence of GS1-compliant data and removes the symbology identifier (if present). For other types of bar codes (UPC-A, UPC-E, EAN-13, EAN-8, ITF-14), an Application Identifier and (in some cases) zeros are added to create a valid GS1 Element String. For example, "01" followed by "0" is prepended to the 13 digits read from an EAN-13 bar code in order to create a GS1 Element String containing a GTIN. The example in Figure 4 shows the symbology identifier being removed from the output of scanning a GS1-128 bar code to yield a GS1 Element String containing a GTIN (AI 01) and serial number (AI 21).

5.1.3. GS1 Element String to "Plain" Syntax

If the business application needs "plain" syntax– for example, if it stores a GTIN and serial number in separate database columns – then the GS1 Element String is parsed to remove the Application Identifiers and to separate multiple data fields. The example in Figure 4 shows the GTIN and serial number being separated from a GS1 Element String in this way.

5.1.4. GS1 Element String to EPC URI

If the business application prefers to use the EPC URI as its application-level representation, the GS1 Element String is translated to an EPC URI. This translation is described in Section 5.5.2.

5.2. Translations from RFID Tag to Business Application

The translations in this section are employed when an RFID tag is read to provide information to a business application and correspond to the upward path along the right slide of Figure 4.

5.2.1. RFID Tag to EPC Binary Encoding

The output from an RFID reader is the binary contents of the memory on the RFID chip, expressed either in binary or, equivalently, in hexadecimal. The contents are communicated through an interface provided by the RFID reader, either a standard interface (e.g., the GS1 Low-Level Reader Protocol, or LLRP), or a reader vendor's proprietary interface. The binary or hexadecimal value includes the full contents of the EPC memory bank of the RFID chip, including control information such as filter values that are embedded in that contents. The output may or may not also include additional control bits such as the Protocol Control (PC) bits or the Extended Protocol Control (XPC) bits. In Figure 4 a 96-bit

hexadecimal value 3074257BF7194E4000001A85 is illustrated, which is the contents of the EPC memory excluding the PC and XPC.

5.2.2. EPC Binary Encoding to EPC Tag URI

The binary or hexadecimal output from the RFID reader is decoded according to the GS1 EPC Tag Data Standard, Section 14, to yield an EPC Tag URI. This is an Internet Uniform Resource Identifier (URI) that represents all data in the EPC portion of the RFID Tag's EPC memory bank, including RFID-specific information such as the filter value. The decoding to the EPC Tag URI may be performed by the hardware or software provided by the RFID reader vendor, or by separate "middleware" software. The example in Figure 4 shows the hexadecimal value being decoded into an SGTIN-96 EPC Tag URI.

5.2.3. EPC Tag URI to Pure Identity EPC URI (aka EPC URI)

The EPC Tag URI contains RFID-specific information such as filter values that are typically only needed at the data capture level. For use at the business application level, the RFID-specific information is stripped away to yield an Pure Identity EPC URI, also called simply an EPC URI. The EPC URI only contains the identification information without any RFID-specific information. This may be performed by the hardware or software provided by the RFID reader vendor or by separate "middleware" software. The example in Figure 4 shows the EPC Tag URI (`urn:epc:tag:…`) converted to an EPC URI using the SGTIN EPC URI scheme (`urn:epc:id:…`), which includes the GTIN (divided into the GS1 Company Prefix and the remainder of the GTIN, but without the check digit) and the serial number.

5.2.4. EPC URI to GS1 Element String

If the business application prefers to use the GS1 Element String as its application-level representation, the EPC URI is translated to a GS1 Element String. This translation is described in Section 5.5.1.

5.3. Business Application to Bar Code

The translations in Sections 5.1 and 5.2 describe the upward direction in Figure 4; that is, when data is read from a data carrier and transmitted "upwards" to a business application. The inverse translations occur in the downward direction: when a business application directs the printing of a bar code or the programming of an RFID Tag. These occur most often at the point in the supply chain where data carriers are affixed for the first time, whether at the point of manufacture or further downstream. These translations are described in this section and in Section 5.4.

The translations in this section are employed when a business application directs the printing of a bar code and correspond to the downward path along the left side of Figure 4.

5.3.1. "Plain" Syntax to GS1 Element String

"Plain" GS1 Identification Keys or other data elements are translated to a GS1 Element string by prefixing each data element with a 2–4 character "Application Identifier" (AI), and then concatenating the results if there is more than one data element. The GS1 General Specifications (Section 3) define the specific AI values and syntax used for each data element, and Section 5.10.2 specifies the rules for concatenating data elements. The example in Figure 4 shows a GTIN and a serial number being combined into a GS1 Element String containing AI 01 and AI 21. For some bar code symbols (UPC-A, UPC-E, EAN-8, EAN-13, and ITF-14) the Application Identifier is implicit in the bar code and so no translation to GS1 Element String is called for.

5.3.2. GS1 Element String to Bar Code Printer Instructions

The interaction between a business application and a bar code printer is not as standardized as the other interactions shown in Figure 4. Typically, the printing device defines a language for describing what is to be printed, including the choice of bar code symbology, the data content for the bar code, physical characteristics of the bar code such as dimensions, human readable characters, etc., and any other printed text or images surrounding the bar code. The GS1 Element String is provided as part of this input. With reference to Figure 4, the GS1 Element String is thus transformed directly into the printed bar code symbol; there is no “Bar Code Reader Output” as there is in the reading direction.

5.4. Business Application to RFID Tag

The translations in this section are employed when a business application directs the programming of an RFID tag and correspond to the downward path along the right slide of Figure 4.

5.4.1. “Plain” Syntax to GS1 Element String

If the business application begins with “plain” syntax, it translates this to a GS1 Element String, as described in Section 5.4.1.

5.4.2. GS1 Element String to Pure Identity EPC URI

If the business application begins with a GS1 Element String as its application-level representation, it translates this to a Pure Identity EPC URI. This translation is described in Section 5.5.2.

5.4.3. Pure Identity EPC URI (aka EPC URI) to EPC Tag URI

The Pure Identity EPC URI contains only the unique identifier that the RFID tag is intended to carry. Additional information must be provided to fully specify what is to be programmed into the RFID Tag’s EPC memory bank. In particular, a choice must be made as to what encoding format is to be used; e.g., for the SGTIN, both a 96-bit format (which restricts the range of serial numbers that may be encoded) and a 198-bit format (which is free of such restrictions but which requires more tag memory) are available. In addition, the RFID tag carries a “filter value” that is sometimes used to assist readers in reading large tag populations efficiently. The EPC Tag URI looks similar to the EPC URI, but with these added pieces of information. The translation process combines the information in the EPC URI with the additional control information in the defined EPC Tag URI syntax. The example in Figure 4 shows an SGTIN EPC URI being translated to an EPC Tag URI by selecting the 96-bit encoding, and adding a filter value of “3”.

5.4.4. EPC Tag URI to EPC Binary Encoding

Unlike a bar code, an RFID Tag contains a digital memory whose content is a string of binary bits. The information in the EPC Tag URI is translated into binary form according to a set of rules defined in the GS1 EPC Tag Data Standard, Section 14. The result is a binary string that can be delivered to an RFID Interrogator or RFID Printer for programming into the tag. The example in Figure 4 shows (in hexadecimal) the binary encoding of the SGTIN EPC Tag URI (`urn:epc:tag:…`) depicted there.

5.5. Translations between Application-level Syntaxes

At the business level, “plain” syntax, GS1 Element String, and EPC URI are available as application-level representations. In some cases a business application needs to translate between these forms. In addition, translations between these forms are needed when transferring data from a bar code to an RFID Tag or vice versa, following an inverted U-shaped path through Figure 4. The relevant

translations are defined in Section 7 of the GS1 EPC Tag Data Standard, and are summarized in the following sections.

5.5.1. Pure Identity EPC URI to GS1 Element String

The Pure Identity EPC URI differs from the GS1 Element string in that the EPC URI uses a few "header" characters to identify the key instead of AI values, and the GS1 Identification Key is separated into a GS1 Company Prefix portion and the remainder of the key (see Section 5.6.4 for special cases without GS1 Company Prefixes). Certain GS1 Identification Keys also have other differences in URI form: e.g., in the case of the GTIN, the EPC URI moves the position of the leading zero or Indicator digit, drops the check digit, and also includes the serial number. To translate from EPC URI to GS1 Element String, the characters of the EPC URI are rearranged according to rules specified in Section 7 of the GS1 EPC Tag Data Standard. In most cases, this also entails the calculation of the check digit part of the GS1 Element String from the other characters of the EPC URI. The example in Figure 4 shows this translation for an SGTIN. Section 7.2 includes an example.

5.5.2. GS1 Element String to Pure Identity EPC URI

In the opposite direction, the AI values in the GS1 Element String are used to determine the EPC URI prefix that is used and the syntax for the remainder of the URI, and rearrangements of characters are reversed compared to the above. The most significant part of this translation is that the digits comprising the GS1 Company Prefix part of the GS1 Identification Key are separated in the EPC URI syntax. This means that it is necessary to know how many characters of the GS1 Identification Key are part of the GS1 Company Prefix. Section 5.6 discusses how this may be done. The example in Figure 4 shows this translation for an SGTIN; in this example, the GS1 Company Prefix is seven digits long. Section 7.1 includes an example.

5.6. Determining the GS1 Company Prefix Length

When translating from a GS1 Element String to the EPC URI, it is necessary to know the length of the GS1 Company Prefix that was used to construct the GS1 Identification Key. This is because the length of the GS1 Company Prefix is variable and these digits must be separated from the remainder of the key to construct the EPC URI. The full procedure for converting a GS1 Element String to a Pure Identity EPC URI is given in Section 7 of the GS1 EPC Tag Data Standard. The conversion rules differ slightly for each type of GS1 Identification Key and each subsection of Section 7 gives the procedure for a particular GS1 Identification Key type. An input to each of these procedures is the length of the GS1 Company Prefix.

The best approach to determine the length of the GS1 Company Prefix depends upon the business context in which the conversion is being performed. The remainder of this section discusses a number of possibilities.

5.6.1. Manual Determination of GS1 Company Prefix Length Using GEPIR

This method employs a manual lookup using the GS1 Global Electronic Party Information Registry (GEPIR) service using GEPIR's web-based interactive user interface. This method is effective when you have a small number of GS1 Element Strings to convert in a process where manual intervention is acceptable. It can also be a useful method for creating a table for use with the methods described in Section 5.6.3.

GEPIR is accessed via a web browser at www.gepir.org. The length of the GS1 Company Prefix can be determined directly by entering a GTIN, an SSCC, or a GLN. For other keys, one of the other methods in this section must be used instead.

Figure 5. GEPIR Main Screen

GEPIR may be used to determine the length of a GS1 Company Prefix in any of the following three ways:

- By entering a GTIN in the “Search by Barcode (GTIN)” screen and selecting the “Trade Item Ownership” option.
- By entering an SSCC in the “Search by Container Code (SSCC)” screen.
- By entering a GLN in the “Search by Location Number (GLN)” screen and selecting the “Owner of GLN” option.

In all cases, if the lookup is successful a small table is displayed. The “GCP” column contains a GS1 Company Prefix. If the value in this column matches the first digits of the key that was entered (disregarding the Indicator digit of a GTIN or the Extension digit of an SSCC) then the number of digits in this value is the length of the GS1 Company Prefix.

If no table is displayed or if the value in the “GCP” column does not match the first digits of the key that was entered, then GEPIR is unable to provide the length of the GS1 Company Prefix for that key.

The above figure and procedure illustrates the use of GEPIR as provided on the GS1 Global Office website. Local GS1 Member Organisations may provide their own interfaces to GEPIR, which may have a different appearance and may use a different language than English.

5.6.2. Programmatic Determination of GS1 Company Prefix Length Using GEPIR Web Services

This method employs a programmatic lookup using the GS1 Global Electronic Party Information Registry (GEPIR) service, using GEPIR’s web services API. This method is effective when you have a

small number of GS1 Element Strings to convert in an automated process. It can also be a useful method for creating a table for use with the methods described in Section 5.6.3.

The GEPIR web service is accessed at the following URL:

http://gepir.gs1.org/v31/router.asmx

Entering this URL into a web browser will display a menu of available web service methods. The `getPartyByGTIN`, `getPartyBySSCC`, and `getPartyByGLN` methods correspond to the three manual lookups described in Section 5.6.2. Each method returns an XML data structure containing an element named `<gcp>`; the value of this element can be interpreted in the same way illustrated in Section 5.6.2 (including the need to confirm that this value matches the first digits of the key). Detailed examples for the web service methods can be found by clicking the individual method names on the page at the above URL, and a formal WSDL specification is available by clicking on the "Service Description" link there.

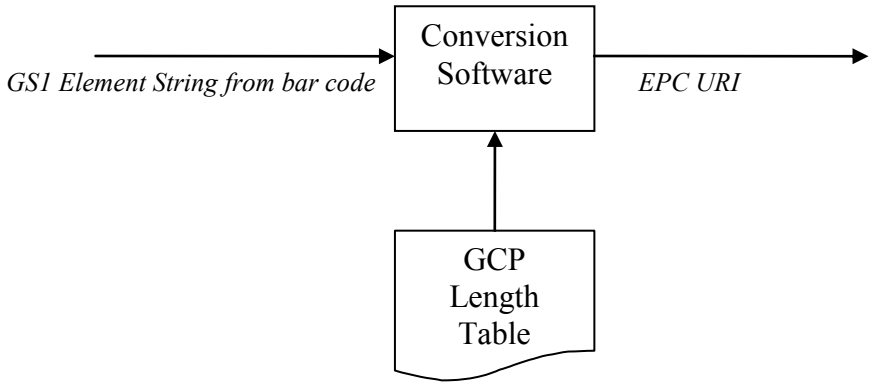
Note that the GEPIR web service may enforce limitations regarding the volume and frequency of invocation, for example, limiting the number of requests per day that a given user may make.

5.6.3. Table-Based Lookup of GS1 Company Prefix Length

In a production setting, it is often necessary to convert many GS1 Element Strings into EPC URIs without human intervention. For example, a receiving process may read serialized GTINs (i.e., GTIN and serial number) from bar codes as goods are received, and generate EPCIS messages containing the EPC URIs corresponding to the bar code contents.

The typical process for this uses conversion software that makes reference to a list of known GS1 Company Prefixes, as illustrated in Figure 6, below:.

Figure 6. Conversion Software's Use of a GCP Length Table



The Conversion Software is an implementation of the translation procedures specified in Section 7 of the GS1 EPC Tag Data Standard. The GCP Length Table is a table that allows the Conversion Software to determine the length of the GS1 Company Prefix.

The simplest form of a GCP Length Table is just a list of GS1 Company Prefixes:

0614141
0623523
0628958
0333333
0823141415
etc.

Given a GS1 Identification Key as input, the Conversion Software determines the length of the GS1 Company Prefix like this:

1. Start with the first six digits of the GS1 Identification Key (skipping the Indicator Digit of a GTIN, the Extension Digit of an SSCC or the zero padding digit of a GRAI). Call this the "candidate prefix".
2. Look up the candidate prefix in the GCP Length Table. If a matching entry has been found, stop: the GS1 Company Prefix has been identified and its length is the length of the candidate prefix.
3. Otherwise, extend the candidate prefix by one digit and repeat from Step 2. Continue until either the length of the GS1 Company Prefix has been found, or the length of the candidate prefix exceeds 12 digits. If the latter, then the GS1 Company Prefix, and therefore its length, cannot be determined by this process.

In this manner, the lookup of any GS1 Identification Key requires at most six probes of the GCP Length Table to determine the length of the GS1 Company Prefix. By using a hash table or similar data structure to store the table, lookup can be done extremely quickly.

A variation on the above idea compresses the data in the table so that only one table entry is needed to represent many GS1 Company Prefixes having the same length and starting with a common set of digits. For example, if it is known that all GS1 Company Prefixes beginning with "06" are 7 digits in length, a length table might look like this:

06 ➔ 7 digits
0333333 ➔ 7 digits
0823141415 ➔ 10 digits

The lookup algorithm above is slightly modified to begin with only one digit in the candidate prefix in order to find entries that have been collapsed as shown above.

There are many commercial software packages that have lookup-based conversion algorithms built in.

The following are recommended approaches for how to obtain a suitable GCP Length Table:

- *Create a List of Your Own GCPs:* If you know that the only GS1 Identification Keys that will be encountered are based on your own GCPs, you can create a list by simply listing all of your own GCPs. This approach typically works well for brand owners and manufacturers who only handle objects that carry GS1 Identification Keys based on their company's GCPs.
- *Create a List of the GCPs of Your Trading Partners:* If you are a downstream party in the supply chain, you may need to convert GS1 Identification Keys containing GCPs that belong to upstream parties. In this case, you may need to include all of your upstream partners' GCPs in your GCP list. One approach to gathering the needed information is to ask your trading partners to provide to you their GCPs as part of your partner on-boarding process.
- *Download a Comprehensive List from GS1*: Certain GS1 Member Organisations have published comprehensive lists of all GS1 Company Prefixes that are registered in their geography. These lists include:
  - http://www.gs1us.org/gcplist - This list contains all GS1 Company Prefixes registered with GS1 US.
  - http://www.onsepc.com/ManagerList.xml - This list contains all GS1 Company Prefixes worldwide that have been registered as "EPC Manager Numbers."

  In addition, work is underway within GS1 to develop a software tool to determine the GS1 Company Prefix length for any GS1 Identification Key, based on a comprehensive lookup table that is expected to use compressed entries as illustrated above. It is envisioned that the GS1 GEPIR network may be leveraged to communicate this information. This work is still in a stage of planning and development.

5.6.4. Special Cases: GTIN-8 and "one off" GS1 Identification Keys

A GTIN-8 does not contain a GS1 Company Prefix. The conversion of a GTIN-8 to a Pure Identity EPC URI is specified in Section 7.1.2 of the GS1 EPC Tag Data Standard. The general rule is that the GTIN-8 is first padded to 14 digits by adding six zeros to the left. Then it is converted to a Pure Identity EPC URI as though it were a GTIN-14 having a GS1 Company Prefix length of eight digits, resulting in `urn:epc:id:sgtin:00000`*`nnn`*`.0`*`nnnn`*`.`*`serial`*.

In some instances, a GS1 Member Organization assigns a "one off" GS1 Identification Key, such as a complete GTIN, GLN, or other key, to a subscribing organization. This situation is dealt with in Section 7 of the GS1 EPC Tag Data Standard, which says the following:

> "In such cases, the GS1 Member Organization holds the GS1 Company Prefix and therefore is responsible for identifying the number of digits that are to occupy the GS1 Company Prefix position within the EPC. The organization receiving the one-off key should consult with its GS1 Member Organization to determine the appropriate number of digits to ascribe to the GS1 Company Prefix portion when constructing a corresponding EPC. In particular, a subscribing organization must not assume that the entire one-off key will occupy the GS1 Company Prefix digits of the EPC, unless specifically instructed by the GS1 Member Organization issuing the key. Moreover, a subscribing organization must not use the digits comprising a particular one-off key to construct any other kind of GS1 Identification Key. For example, if a subscribing organization is issued a one-off GLN, it must not create SSCCs using the 12 digits of the one-off GLN as though it were a 12-digit GS1 Company Prefix."

For example, if a lookup table as described in Section 5.6.3 is used, the entries for one-off keys should reflect the MO's determination of what GS1 Company Prefix length to ascribe to those keys.

6. Scenarios

This section illustrates complete business scenarios in which bar code / RFID interoperability issues arise.

6.1. Serialization of Pharmaceuticals Using Bar Code and RFID

In this scenario, a US pharmaceutical manufacturer plans to serialize each drug package using a GTIN plus serial number by applying both a GS1 DataMatrix 2D bar code and an EPC RFID tag on each drug package. (Note that the details would be nearly identical if a different type of bar code were used, such as a GS1-128 or GS1 DataBar.)

6.1.1. Choices and Decisions

Because this manufacturer plans to make use of two carrier technologies on each drug package it is very important for them to first define the necessary GS1 Identification Keys in one of the application-level forms (Table 1 in Section 3.1) and only then convert the keys into the forms necessary for each of the two data carriers. Because RFID is one of the two data carriers to be used in this scenario, the Pure Identity EPC URI is a good choice for the application-level form. This is because the Pure Identity EPC URI application-level form ensures that the length of the GS1 Company Prefix (GCP) is established from the beginning and will be available to the Data Capture Workflow layer that exists within the Data Capture Application (see Figure 3 in Section 4) that will need to construct the carrier-specific encodings of the GS1 Identification Keys.

The GS1 Identification Keys this manufacturer has decided to apply to each drug package are a GTIN—which will be the same on each drug package according to GTIN allocation rules—and a serial number—which will be unique to each drug package for that GTIN. In the Pure Identity EPC URI form this is known as a Serialized GTIN (SGTIN) and takes the following form

urn:epc:id:sgtin:*<GCP>*.*<indicator digit+ Item Reference>*.*<serial number>*

Refer to the GS1 EPC Tag Data Standard for the full details of constructing an EPC URI.

Other data elements, such as lot and expiration date, may also be included in both data carriers but this is beyond the scope of this scenario. See the GS1 EPC Tag Data Standard and the GS1 General Specifications for details on adding related data elements.

The manufacturer could choose to generate and store their SGTIN and visibility data in many different ways. Because they have already chosen to use the Pure Identity EPC URI for their application-level form of the SGTINs they will most likely choose to use a data repository and packaging solution that is based on the GS1 EPCIS standard. EPCIS-based applications work well for all serialization applications whether they are based on bar codes alone, RFID alone, or a combination of both. The SGTIN EPC URI form can be used directly in that type of system.

6.1.2. Encoding

When it is time to assign a specific serialized GTIN to a specific drug package, the Data Capture Workflow must convert each SGTIN Pure Identity EPC URI into two different carrier-specific forms. It then communicates with external devices on the packaging line to ensure that the same SGTIN is encoded into a GS1 DataMatrix bar code and programmed into a EPC RFID tag and that both are then applied to the same drug package. There are several approaches to accomplishing the complex material handling requirements in this step but they are not within the scope of this document.

To properly encode the SGTIN in a GS1 DataMatrix bar code the Data Capture Workflow must convert the SGTIN from the Pure Identity EPC URI form into the GS1 Element String form (see Section 5) and then encode with additional characters necessary to cause the bar code printer to print a GS1 DataMatrix. To properly encode the same SGTIN in a EPC RFID tag the Data Capture Workflow must convert the SGTIN from the Pure Identity EPC URI form into the EPC Tag URI form (see Section 5) that matches the type of physical RFID tags that are being used. The Data Capture Workflow must then transmit this form to the RFID writer/reader through interfaces (e.g., ALE and LLRP; see Section 4).

6.1.3. Data Verification

Serialization-based supply chain applications rely on the fact that each unit will contain a serial number that is unique from all other units. For a complex application like this where the same SGTIN serial number is intended to be carried in two different data carriers on the same package, it would be a major error if the two data carriers on a given package ended up containing different SGTINs. To ensure that errors like this are detected and prevented from entering the supply chain it would typically be necessary to confirm that both of the data carriers applied to each drug package contain the same SGTIN as part of a quality process before the package is shipped. In this verification process an RFID reader and a 2D bar code reader would be positioned so that one drug package at a time passes both readers. Both devices would read the contents of the respective data carrier on one package and this data would be transmitted through the appropriate interfaces to the Data Capture Workflow application.

Because these two data streams are carrier-specific, they cannot be compared directly. One way to compare them would be to use the Data Capture Workflow application to convert both of them back to one of the application-level forms. Finally, these two values can be compared directly to confirm that the same SGTIN was read from both of the data carriers applied to the same drug package.

6.1.4. Data Sharing

Because the manufacturer wants their customers to be able to read the SGTIN from either the EPC RFID tag or the GS1 DataMatrix bar code and then interpret it properly, they may transmit the SGTINs on each package shipped to a given customer in one of the application-level forms. Because RFID is

one of the two data carriers, the Pure Identity EPC URI may be the preferred form because it will eliminate the need for the customer to obtain the GCP lengths in advance since the GCP is fully delineated in the EPC URI application-level form. This can be done by passing EPCIS Commissioning and Shipping events (which by definition will include the SGTIN in EPC URI form) to the customer, or by passing GS1 EANCOM Despatch Advice Notice (DESADV) messages that contain the GTINs and serial numbers in "plain" syntax. Other business process considerations will dictate whether EPCIS events, DESADV messages, or both, are required, as they serve different purposes.

6.1.5. Receipt Confirmation

When the manufacturer's customer receives the physical shipment they can confirm that they received the drug packages that exactly match those that the manufacturer listed in their electronic message (either EPCIS events or EDI message) by reading the SGTIN from either the EPC RFID tag or the GS1 DataMatrix bar code. This confirmation may be desirable for Track & Trace or ePedigree applications. Or the customer may simply wish to read the SGTINs of the drug packages they received so that they can keep a record of them.

To confirm that the SGTINs received match those that were shipped by the manufacturer, the Data Capture Workflow application within the customer's system will need to convert the carrier-specific form read by either the EPC RFID reader or the 2D bar code reader to one of the application-level forms. The form chosen by the customer may be guided by the form that the manufacturer provided in advance, or the form that the customer has stored that data, or whatever form in which the customer prefers to hold that data going forward. For confirmation purposes, the customer will convert the SGTINs they read to the form that matches the data they received from the manufacturer. The customer's application can then use the SGTINs read from the product to find a matching SGTIN from the electronic communication received from the manufacturer. If there is a matching SGTIN found, the customer has confirmed that the manufacturer intended to include this particular drug package in the received shipment.

Whether choosing to perform this comparison or not, the customer may wish to keep a record of all SGTINs received from the manufacturer in their own EPCIS-based repository. In that case, the EPC URI syntax would be used.

6.2. Labelling of a Logistic Unit Using Bar Code and RFID

In this scenario, a distributor wishes to identify a logistic unit using a Serial Shipping Container Code (SSCC) as assigned by its Warehouse Management System (WMS) software and mark the logistic unit for shipping using both a GS1 Logistics Label containing a GS1-128 bar code and an EPC RFID Tag.

6.2.1. Choices and Decisions

Because this distributor plans to make use of two carrier technologies on each logistic unit it is very important for them to first define the necessary GS1 Identification Keys in one of the application-level forms (Table 1 in Section 3.1) and only then convert the keys into the forms necessary for each of the two data carriers. In this scenario, the Warehouse Management System uses "plain" syntax to represent the SSCC, which is an 18-character numeric string.

6.2.2. Encoding

When it is time to label a logistic unit, the Data Capture Workflow must convert the SSCC for the logistic unit as received from the WMS into two different carrier-specific forms. It then communicates with external devices on the warehouse floor to ensure that the same SSCC is encoded into a GS1-128 bar code and programmed into a EPC RFID tag that are part of the same label.

To properly encode the SSCC in a GS1-128 bar code the Data Capture Workflow must convert the 18-digit SSCC into the GS1 Element String form (see Section 5) by prefixing with the Application Identifier "00", and then encode with additional characters necessary to cause the bar code printer to print a GS1-128 symbol containing that GS1 Element String. To properly encode the same SSCC in a EPC RFID tag the Data Capture Workflow must convert the SSCC into the binary form for programming into the RFID tag. This is done in several steps, each described in Section 5:

- The Data Capture Workflow converts the SSCC into an EPC Pure Identity URI. In carrying out this step, the Data Capture Workflow must determine the length of the GS1 Company Prefix within the SSCC, using any of the techniques defined in Section 5.6.
- The Data Capture Workflow converts the Pure Identity EPC URI form into the EPC Tag URI form (see Section 5) for a 96-bit RFID tag.
- The Data Capture Workflow must then transmit this form to the RFID writer/reader through interfaces (e.g., ALE and LLRP; see Section 4), which in turn converts the EPC Tag URI to binary form and programs the tag.

6.3. Serialization of a Reusable Asset Using Bar Code and RFID

In this scenario, a service provider supplies reusable plastic totes to end users in the food supply chain. The service provider wishes to assign a unique Global Returnable Asset Identifier (GRAI) to each tote, and carry this code both on a GS1-128 bar code label and an embedded 96-bit RFID tag.

6.3.1. Choices and Decisions

Because this service provider plans to make use of two carrier technologies on each tote it is very important for them to first define the necessary GS1 Identification Keys in one of the application-level forms (Table 1 in Section 3.1) and only then convert the keys into the forms necessary for each of the two data carriers. Because RFID is one of the two data carriers to be used in this scenario, the Pure Identity EPC URI is a good choice for the application-level form.

The GS1 Identification Key in this application is the Global Returnable Asset Identifier (GRAI), which in Pure Identity EPC URI form appears as follows:

`urn:epc:id:grai:`*<GCP>*`.`*<Asset Type>*`.`*<serial number>*

Refer to the GS1 Tag Data Standard for the full details on constructing an EPC URI.

6.3.2. Encoding

When it is time to assign a specific GRAI to a specific tote, the Data Capture Workflow must convert each Pure Identity EPC URI GRAI into two different carrier-specific forms and communicate with external devices on the tote preparation line to ensure that the same GRAI is encoded into a GS1-128 bar code and programmed into a EPC RFID tag and that both are then applied to the same tote.

To properly encode the GRAI in a GS1-128 bar code the Data Capture Workflow must convert the GRI from the Pure Identity EPC URI form into the GS1 Element String form (see Section 5) and then encode with additional characters necessary to cause the bar code printer to print a GS1-128 symbol. To properly encode the same GRAI in an EPC RFID tag the Data Capture Workflow must convert the GRAI from the Pure Identity EPC URI form into the EPC Tag URI form (see Section 5) that matches the type of physical RFID tags that are being used. The Data Capture Workflow must then transmit this form to the RFID writer/reader through interfaces (e.g., ALE and LLRP; see Section 4), which in turn converts the EPC Tag URI to binary form and programs the tag.

7. Appendix: Encoding/Decoding Illustrations

This section presents six detailed examples showing encoding and decoding between GS1 Identification Keys (SGTIN, SSCC, and GRAI) and the EPC memory bank of a Gen 2 RFID tag.

As these are merely illustrative examples, in all cases the indicated normative sections of the GS1 EPC Tag Data Standard should be consulted for the definitive rules for encoding and decoding. The diagrams and accompanying notes in this section are not intended to be a complete specification for encoding or decoding, but instead serve only to illustrate the highlights of how the normative encoding and decoding procedures function. The procedures for encoding other types of identifiers are different in significant ways, and the appropriate sections of the GS1 EPC Tag Data Standard should be consulted.

7.1. Encoding a Serialized Global Trade Item Number (SGTIN) to SGTIN-96

[This section reproduces Section E.1 of the GS1 EPC Tag Data Standard.]

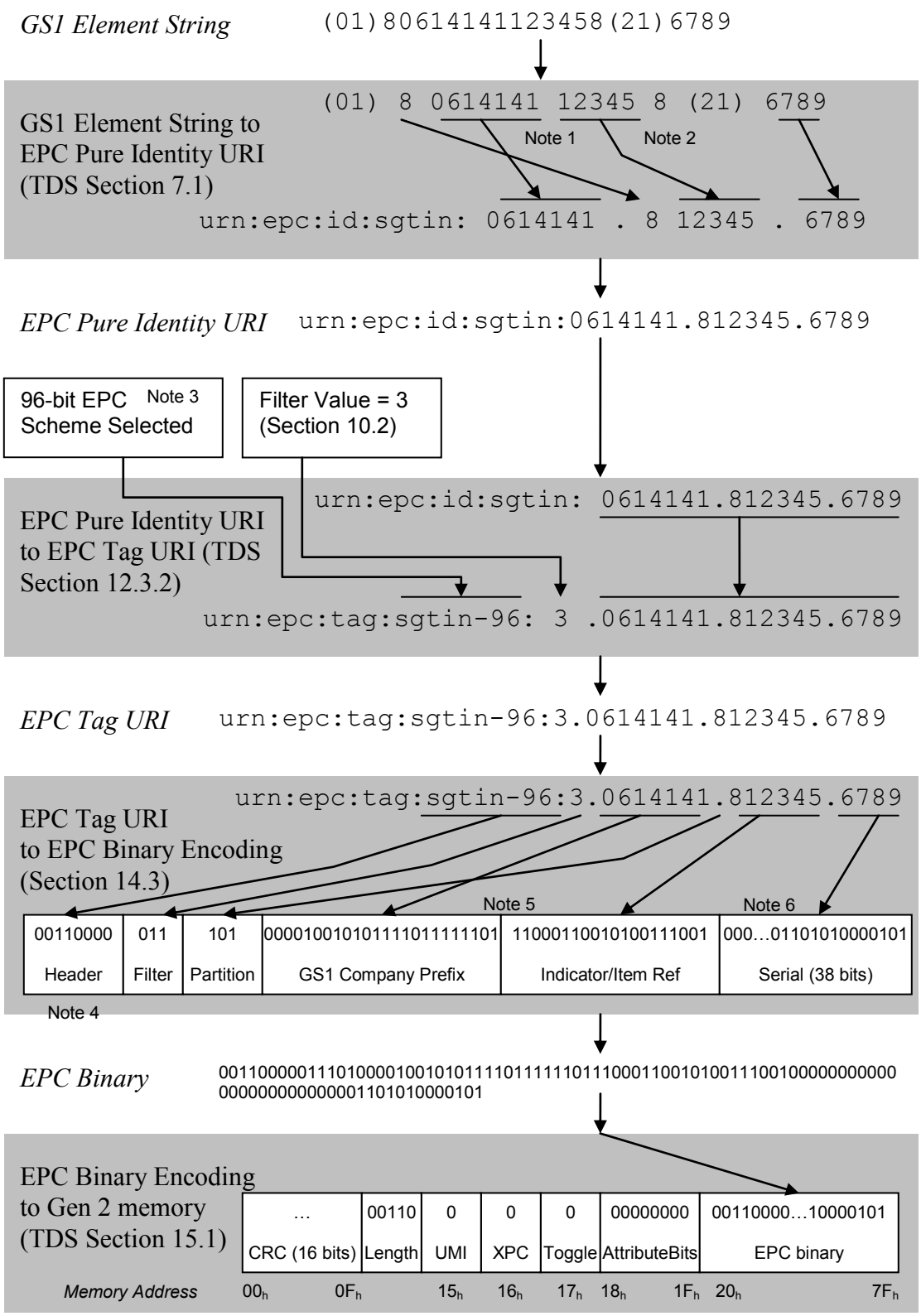
This example illustrates the encoding of a GS1 Element String containing a Serialized Global Trade Item Number (SGTIN) into an EPC Gen 2 RFID tag using the SGTIN-96 EPC scheme, with intermediate steps including the EPC URI, the EPC Tag URI, and the EPC Binary Encoding.

In some applications, only a part of this illustration is relevant. For example, an application may only need to transform a GS1 Element String into an EPC URI, in which case only the top of the illustration is needed.

Section numbers in the illustration below refer to sections of the GS1 EPC Tag Data Standard. The illustration below makes reference to the following notes:

- Note 1: The step of converting a GS1 Element String into the EPC Pure Identity URI requires that the number of digits in the GS1 Company Prefix be determined; e.g., by reference to an external table of company prefixes. In this example, the GS1 Company Prefix is shown to be seven digits.
- Note 2: The check digit in GTIN as it appears in the GS1 Element String is not included in the EPC Pure Identity URI.
- Note 3: The SGTIN-96 EPC scheme may only be used if the Serial Number meets certain constraints. Specifically, the serial number must (a) consist only of digit characters; (b) not begin with a zero digit (unless the entire serial number is the single digit '0'); and (c) correspond to a decimal numeral whose numeric value that is less than 2^{38} (less than 274,877,906,944). For all other serial numbers, the SGTIN-198 EPC scheme must be used. Note that the EPC URI is identical regardless of whether SGTIN-96 or SGTIN-198 is used in the RFID Tag.
- Note 4: EPC Binary Encoding header values are defined in Section 14.2 of the GS1 EPC Tag Data Standard.
- Note 5: The number of bits in the GS1 Company Prefix and Indicator/Item Reference fields in the EPC Binary Encoding depends on the number of digits in the GS1 Company Prefix portion of the EPC URI, and this is indicated by a code in the Partition field of the EPC Binary Encoding. See Table 17 of the GS1 EPC Tag Data Standard (for the SGTIN EPC only).
- Note 6: The Serial field of the EPC Binary Encoding for SGTIN-96 is 38 bits; not all bits are shown here due to space limitations.

Figure 7. Encoding a Serialized Global Trade Item Number (SGTIN) to SGTIN-96



7.2. Decoding an SGTIN-96 to a Serialized Global Trade Item Number (SGTIN)

[This section reproduces Section E.2 of the GS1 EPC Tag Data Standard.]

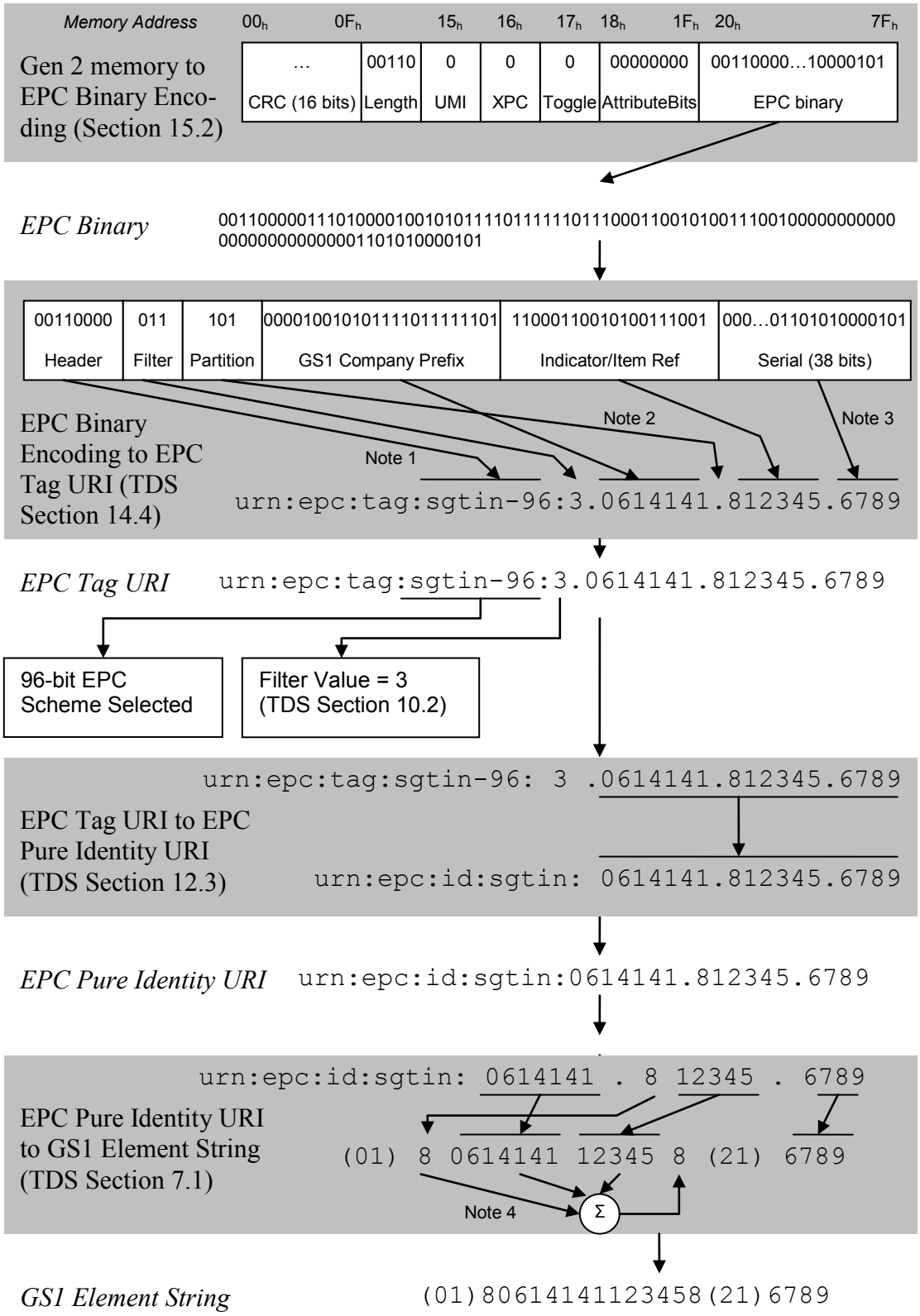
This example illustrates the decoding of an EPC Gen 2 RFID tag containing an SGTIN-96 EPC Binary Encoding into a GS1 Element String containing a Serialized Global Trade Item Number (SGTIN), with intermediate steps including the EPC Binary Encoding, the EPC Tag URI, and the EPC URI.

In some applications, only a part of this illustration is relevant. For example, an application may only need to decode RFID tag contents into an EPC URI, in which case only the top of the illustration is needed.

Section numbers in the illustration below refer to sections of the GS1 EPC Tag Data Standard. The illustration below makes reference to the following notes:

- Note 1: The EPC Binary Encoding header indicates how to interpret the remainder of the binary data, and the EPC scheme name to be included in the EPC Tag URI. EPC Binary Encoding header values are defined in Section 14.2 of the GS1 EPC Tag Data Standard.
- Note 2: The Partition field of the EPC Binary Encoding contains a code that indicates the number of bits in the GS1 Company Prefix field and the Indicator/Item Reference field. The partition code also determines the number of decimal digits to be used for those fields in the EPC Tag URI (the decimal representation for those two fields is padded on the left with zero characters as necessary). See Table 17 of the GS1 EPC Tag Data Standard (for the SGTIN EPC only).
- Note 3: For the SGTIN-96 EPC scheme, the Serial Number field is decoded by interpreting the bits as a binary integer and converting to a decimal numeral without leading zeros (unless all serial number bits are zero, which decodes as the string "0"). Serial numbers containing non-digit characters or that begin with leading zero characters may only be encoded in the SGTIN-198 EPC scheme.
- Note 4: The check digit in the GS1 Element String is calculated from other digits in the EPC Pure Identity URI, as specified in Section 7.1 of the GS1 EPC Tag Data Standard.

Figure 8. Decoding an SGTIN-96 to a Serialized Global Trade Item Number (SGTIN)



7.3. Encoding a Serial Shipping Container Code (SSCC) to SSCC-96

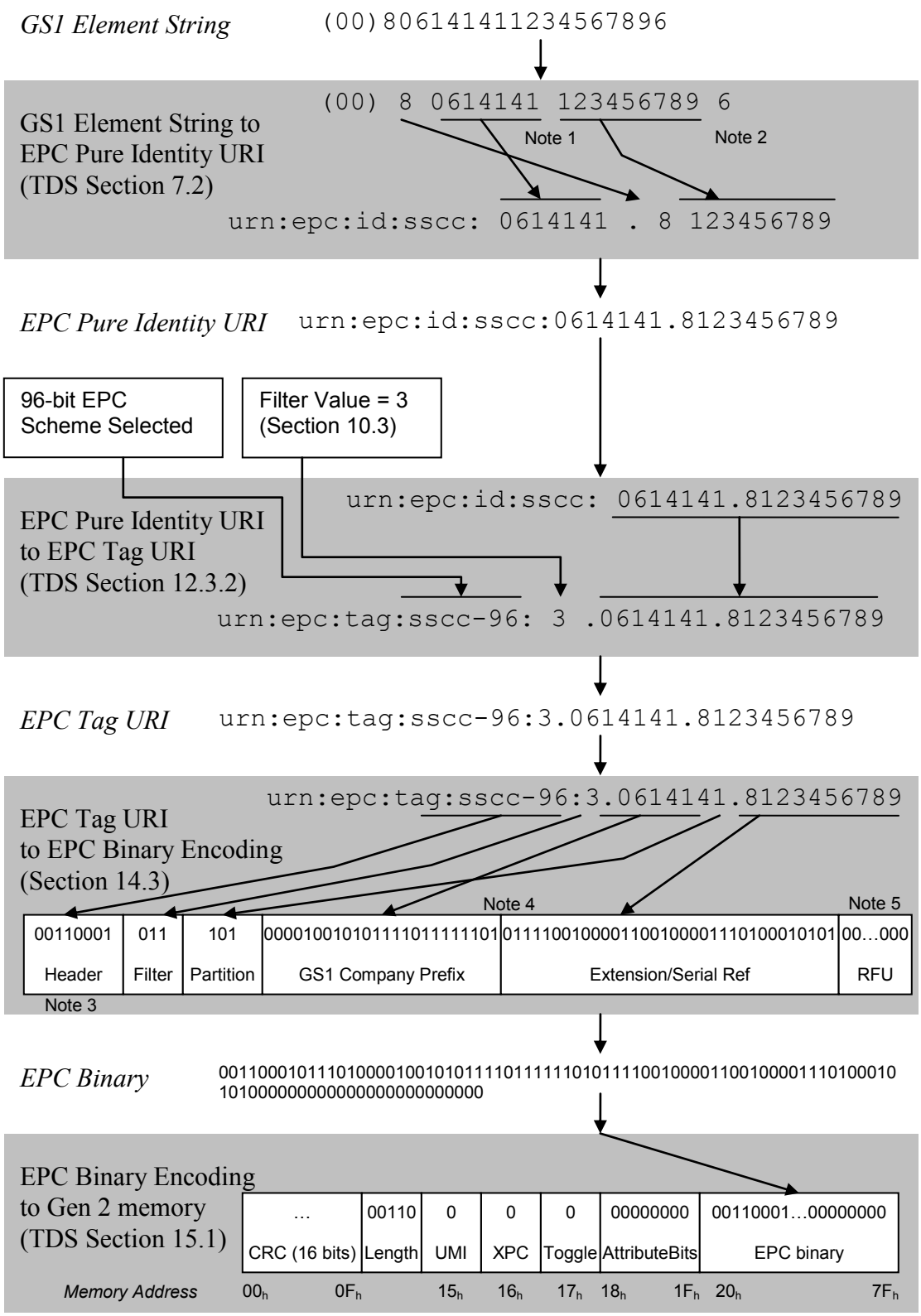
This example illustrates the encoding of a GS1 Element String containing a Serial Shipping Container Code (SSCC) into an EPC Gen 2 RFID tag using the SSCC-96 EPC scheme, with intermediate steps including the EPC URI, the EPC Tag URI, and the EPC Binary Encoding.

In some applications, only a part of this illustration is relevant. For example, an application may only need to transform a GS1 Element String into an EPC URI, in which case only the top of the illustration is needed.

Section numbers in the illustration below refer to sections of the GS1 EPC Tag Data Standard. The illustration below makes reference to the following notes:

- Note 1: The step of converting a GS1 Element String into the EPC Pure Identity URI requires that the number of digits in the GS1 Company Prefix be determined; e.g., by reference to an external table of company prefixes. In this example, the GS1 Company Prefix is shown to be seven digits.
- Note 2: The check digit in SSCC as it appears in the GS1 Element String is not included in the EPC Pure Identity URI.
- Note 3: EPC Binary Encoding header values are defined in Section 14.2 of the GS1 EPC Tag Data Standard.
- Note 4: The number of bits in the GS1 Company Prefix and Extension/Serial Reference fields in the EPC Binary Encoding depends on the number of digits in the GS1 Company Prefix portion of the EPC URI, and this is indicated by a code in the Partition field of the EPC Binary Encoding. See Table 20 of the GS1 EPC Tag Data Standard (for the SSCC EPC only).
- Note 5: The final 24 of the EPC Binary Encoding for SSCC-96 are reserved and must be all zeros; not all bits are shown here due to space limitations.

Figure 9. Encoding a Serial Shipping Container Code (SSCC) to SSCC-96



7.4. Decoding an SSCC-96 to a Serial Shipping Container Code (SSCC)

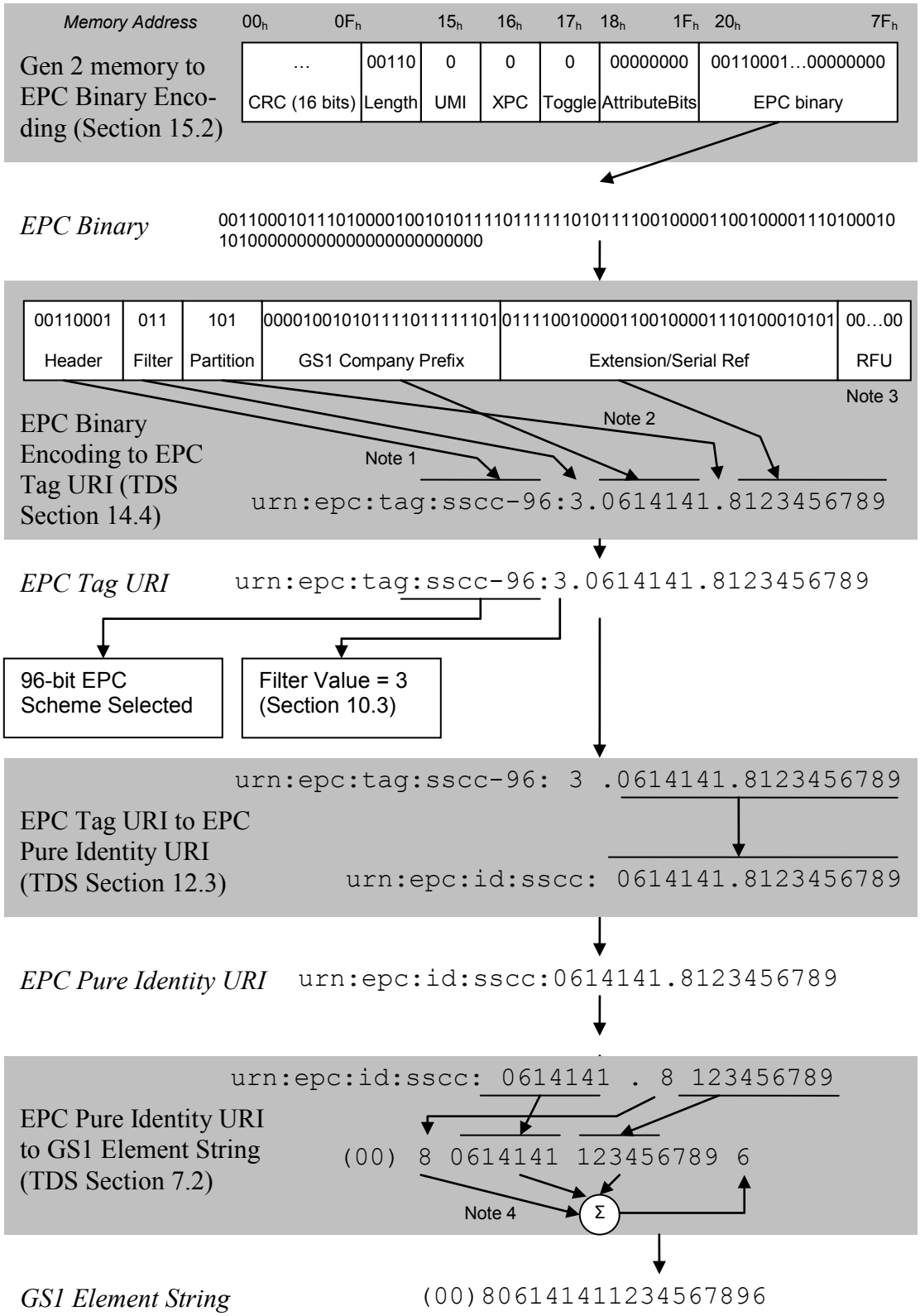
This example illustrates the decoding of an EPC Gen 2 RFID tag containing an SSCC-96 EPC Binary Encoding into a GS1 Element String containing a Serial Shipping Container Code (SSCC), with intermediate steps including the EPC Binary Encoding, the EPC Tag URI, and the EPC URI.

In some applications, only a part of this illustration is relevant. For example, an application may only need to decode RFID tag contents into an EPC URI, in which case only the top of the illustration is needed.

Section numbers in the illustration below refer to sections of the GS1 EPC Tag Data Standard. The illustration below makes reference to the following notes:

- Note 1: The EPC Binary Encoding header indicates how to interpret the remainder of the binary data, and the EPC scheme name to be included in the EPC Tag URI. EPC Binary Encoding header values are defined in Section 14.2 of the GS1 EPC Tag Data Standard.
- Note 2: The Partition field of the EPC Binary Encoding contains a code that indicates the number of bits in the GS1 Company Prefix field and the Extension/Serial Reference field. The partition code also determines the number of decimal digits to be used for those fields in the EPC Tag URI (the decimal representation for those two fields is padded on the left with zero characters as necessary). See Table 20 of the GS1 EPC Tag Data Standard (for the SSCC EPC only).
- Note 3: The final 24 of the EPC Binary Encoding for SSCC-96 are reserved and must be all zeros; not all bits are shown here due to space limitations.
- Note 4: The check digit in the GS1 Element String is calculated from other digits in the EPC Pure Identity URI, as specified in Section 7.2 of the GS1 EPC Tag Data Standard.

Figure 10. Decoding an SSCC-96 to a Serial Shipping Container Code (SSCC)



7.5. Encoding a Global Returnable Asset Identifier (GRAI) to GRAI-96

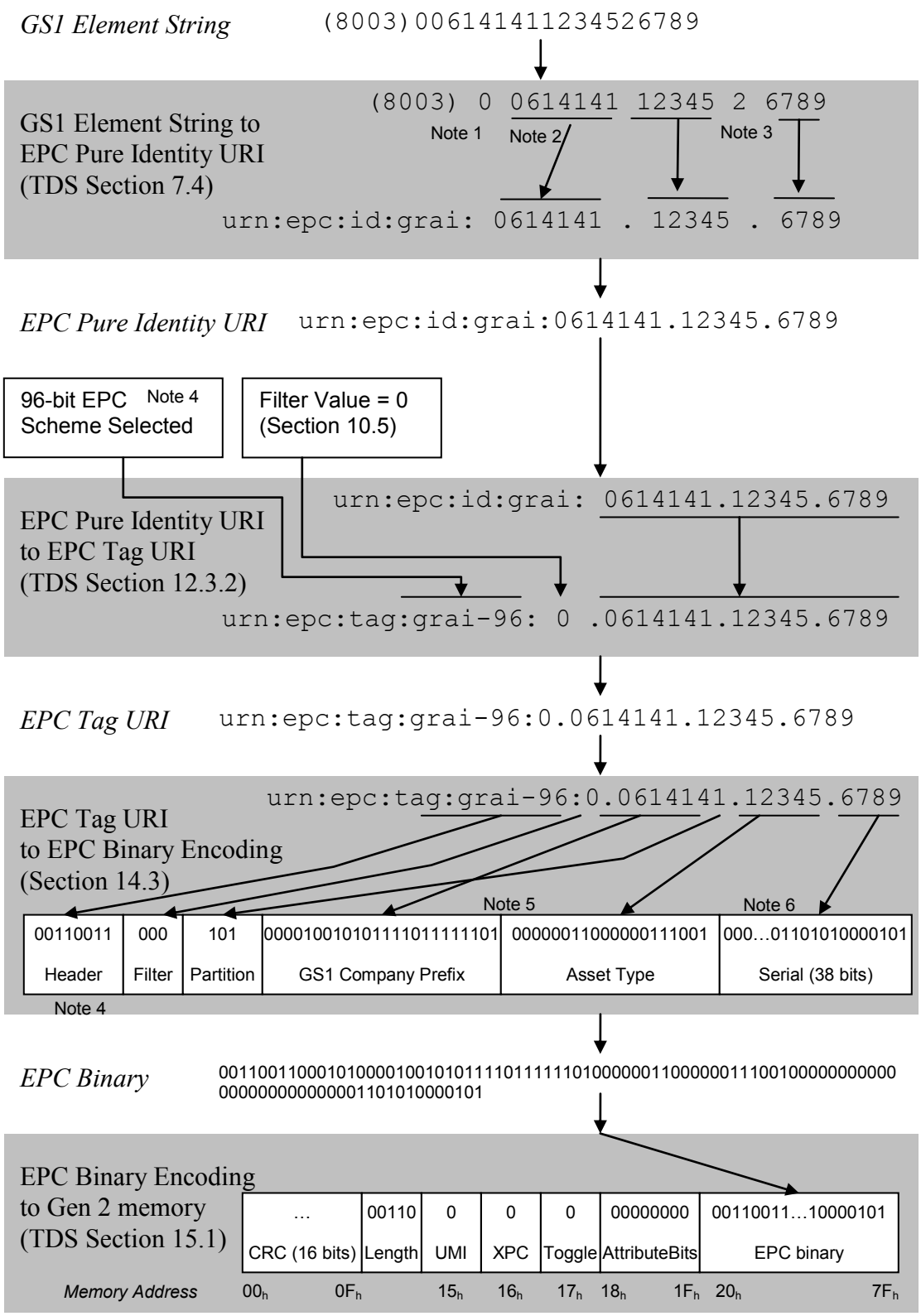
This example illustrates the encoding of a GS1 Element String containing a Global Returnable Asset Identifier (GRAI), including a serial number, into an EPC Gen 2 RFID tag using the GRAI-96 EPC scheme, with intermediate steps including the EPC URI, the EPC Tag URI, and the EPC Binary Encoding.

In some applications, only a part of this illustration is relevant. For example, an application may only need to transform a GS1 Element String into an EPC URI, in which case only the top of the illustration is needed.

Section numbers in the illustration below refer to sections of the GS1 EPC Tag Data Standard. The illustration below makes reference to the following notes:

- Note 1: The GS1 Element String for the GRAI contains a single zero "filler" digit between the AI (8003) and the digits of the GRAI. This zero filler is not considered part of the GRAI, and does not appear in a "plain" GRAI key, nor in the EPC Pure Identity URI
- Note 2: The step of converting a GS1 Element String into the EPC Pure Identity URI requires that the number of digits in the GS1 Company Prefix be determined; e.g., by reference to an external table of company prefixes. In this example, the GS1 Company Prefix is shown to be seven digits.
- Note 3: The check digit in GRAI as it appears in the GS1 Element String is not included in the EPC Pure Identity URI.
- Note 4: The GRAI-96 EPC scheme may only be used if the Serial Number meets certain constraints. Specifically, the serial number must (a) consist only of digit characters; (b) not begin with a zero digit (unless the entire serial number is the single digit '0'); and (c) correspond to a decimal numeral whose numeric value that is less than 2^{38} (less than 274,877,906,944). For all other serial numbers, the GRAI-170 EPC scheme must be used. Note that the EPC URI is identical regardless of whether GRAI-96 or GRAI-170 is used in the RFID Tag.
- Note 5: EPC Binary Encoding header values are defined in Section 14.2 of the GS1 EPC Tag Data Standard.
- Note 6: The number of bits in the GS1 Company Prefix and Asset Type fields in the EPC Binary Encoding depends on the number of digits in the GS1 Company Prefix portion of the EPC URI, and this is indicated by a code in the Partition field of the EPC Binary Encoding. See Table 25 of the GS1 EPC Tag Data Standard (for the GRAI EPC only).
- Note 7: The Serial field of the EPC Binary Encoding for GRAI-96 is 38 bits; not all bits are shown here due to space limitations.

Figure 11. Encoding a Global Returnable Asset Identifier (GRAI) to GRAI-96



7.6. Decoding a GRAI-96 to a Global Returnable Asset Identifier (GRAI)

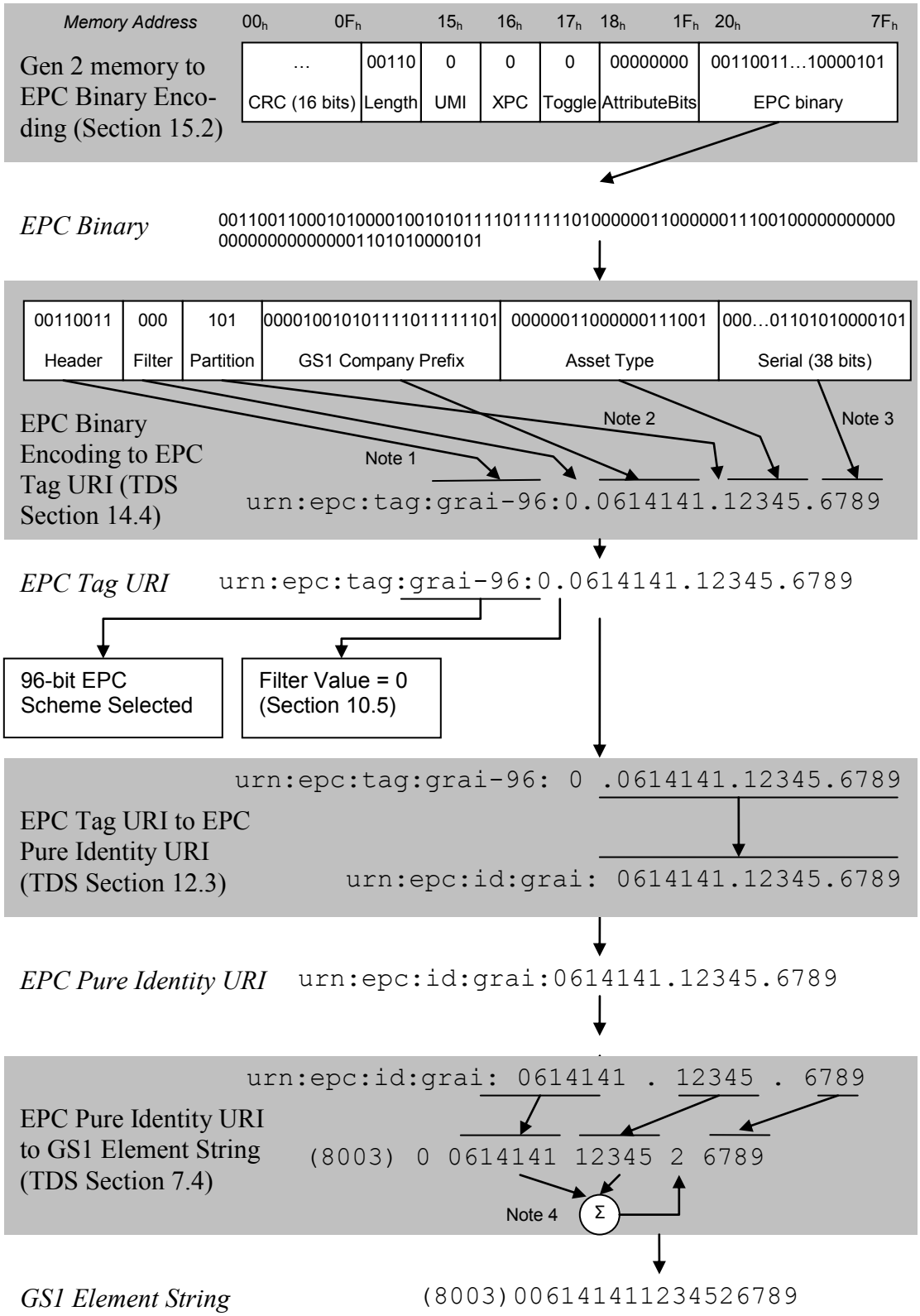
This example illustrates the decoding of an EPC Gen 2 RFID tag containing a GRAI-96 EPC Binary Encoding into a GS1 Element String containing a Global Returnable Asset Identifier (GRAI), including serial number, with intermediate steps including the EPC Binary Encoding, the EPC Tag URI, and the EPC URI.

In some applications, only a part of this illustration is relevant. For example, an application may only need to decode RFID tag contents into an EPC URI, in which case only the top of the illustration is needed.

Section numbers in the illustration below refer to sections of the GS1 EPC Tag Data Standard. The illustration below makes reference to the following notes:

- Note 1: The EPC Binary Encoding header indicates how to interpret the remainder of the binary data, and the EPC scheme name to be included in the EPC Tag URI. EPC Binary Encoding header values are defined in Section 14.2 of the GS1 EPC Tag Data Standard.
- Note 2: The Partition field of the EPC Binary Encoding contains a code that indicates the number of bits in the GS1 Company Prefix field and the Asset Type field. The partition code also determines the number of decimal digits to be used for those fields in the EPC Tag URI (the decimal representation for those two fields is padded on the left with zero characters as necessary). See Table 25 of the GS1 EPC Tag Data Standard (for the GRAI EPC only).
- Note 3: For the GRAI-96 EPC scheme, the Serial Number field is decoded by interpreting the bits as a binary integer and converting to a decimal numeral without leading zeros (unless all serial number bits are zero, which decodes as the string "0"). Serial numbers containing non-digit characters or that begin with leading zero characters may only be encoded in the GRAI-170 EPC scheme.
- Note 4: The check digit in the GS1 Element String is calculated from other digits in the EPC Pure Identity URI, as specified in Section 7.4 of the GS1 EPC Tag Data Standard.

Figure 12. Decoding an SGTIN-96 to a Serialized Global Trade Item Number (SGTIN)



8. Appendix: Summary of GS1 Data Carriers

The following table lists all GS1 Data Carriers and the data they may contain in accordance with the GS1 General Specifications.

Table 7. Data Content of GS1 Data Carriers

GS1 Data Carrier	GS1 Key	GS1 Key Attributes	Notes
EAN-13	GTIN-13	None	
UPC-A	GTIN-12	None	
EAN-8	GTIN-8	None	
UPC-E	GTIN-12	None	Zero-suppressible GTIN-12s only
ITF-14	GTIN	None	
GS1-128	All	All	
GS1 DataBar Expanded	GTIN	Trade Item AIs	
GS1 DataBar Limited	GTIN	Trade Item AIs	Only indicator digit 1
All other GS1 Data Bar	GTIN	Trade Item AIs	
GS1 DataMatrix	All	All	In addition to trade item ID & marking at various packaging levels includes GTIN+Serial, and potential backward compatible use of GIAI or GRAI, on medical devices via Direct Part Mark
GS1 QR Code	GTIN	Trade Item AIs	AI (8200) mandatory
GS1 Composite Component	None	All	Used only with Regulated Healthcare GTIN
GS1 RFID (UHF and HF Gen 2)	GTIN (serial required), SSCC, GLN (with or without extension), GRAI (serial required), GIAI, GDTI (serial required), GSRN	All (in user memory)	RFID tags with 96 bits of EPC memory impose restrictions on GTIN serial, GRAI serial, GIAI serial, GDTI serial, and GLN extension

The following table lists all GS1 bar codes and their carrier-specific functional elements.

Table 8. Carrier-specific Functional Element of GS1 Bar Codes

GS1 Data Carrier	Symbology Identifier	GS1 Start Pattern or Mandatory Lead Character	Delimiter	Data Element Qualifier
EAN-13	]E0	guard bars	N/A	N/A
UPC-A	]E0	guard bars	N/A	N/A
EAN-8	]E4	guard bars	N/A	N/A
UPC-E	]E0	guard bars	N/A	N/A
ITF-14	]I1	I-2/5 Start Pattern	N/A	N/A
GS1-128	]C1	FNC1	FNC1	AI
GS1 DataBar Expanded	]e0	guard pattern	FNC1	AI
GS1 DataBar Limited	]e0	guard pattern	N/A	AI 01
All other GS1 Data Bar	]e0	guard pattern	N/A	AI 01
GS1 DataMatrix	]d2	FNC1	FNC1	AI
GS1 QR Code	]Q3	FNC1 (mode)	FNC1 (alias)	AI
GS1 Composite Component	]e1 /]e2	row address pattern	FNC1	AI

Carrier-specific functional elements of GS1 RFID data carriers (UHF and HF Gen 2) are specified in the GS1 EPC Tag Data Standard.

9. Appendix: References

- GS1 EPC Tag Data Standard 1.6, September 2011, http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_6-RatifiedStd-20110922.pdf
- GS1 General Specifications Version 12, January 2012, http://www.gs1.org/barcodes/technical/genspecs