The Global Language of Business

GS1 Digital Link: Compression

enabling consistent representation of GS1 identification keys within web addresses to link to online information and services

To Be 1.2, Draft 0.5, 19 November 2020

Document Summary

Document Item	Current Value
Document Name	GS1 Digital Link: Compression
Document Date	19 November 2020
Document Version	To be 1.2
Document Issue	
Document Status	Draft
Document Description	enabling consistent representation of GS1 identification keys within web addresses to link to online information and services

Contributors

Name	Company
Kishore Karuppan (Chair)	Procter & Gamble Co.
Michel Ottiker (Chair)	GS1 Switzerland
Dominique Guinard (Chair)	EVRYTHNG
Stefan Artlich	Bayer AG - Division Pharma
Adam Björnstjerna	HKScan Sweden AB
Wes Bloemker	Arthrex Inc.
Stephen Brown	Mead Westvaco
Greg Buckley	PepsiCo, Inc.
Jeanne Duckett	Avery Dennison RFID
Vera Feuerstein	Nestlé
Tarik Gemei	Amgen Inc.
Viktoria Grahnen	AbbVie
Zoltan Homan	Cook Medical Inc.
Sinead Kennedy	Cook Medical Australia
Mike Kuhno	Avery Dennison RFID
Nicolas Lecocq	L'Oreal
Phill Marley	AstraZeneca Pharmaceuticals
FRITZ MBUMB-KUMB	EM Microelectronic
Yi Meng	Qingdao Haier Washing Machine Co. Ltd.
Paul Muller	EM Microelectronic
Martin Neselius	BillerudKorsnäs (Venture)
Tatjana Pathare	F. Hoffmann-La Roche Ltd.
Mandeep Sodhi	Nestlé
Jim Springer	EM Microelectronic
John Terwilliger	Abbott
Gina Tomassi	PepsiCo, Inc.
Jeroen van Rosmalen	Amgen Inc.
Julie Vargas	Avery Dennison RFID
Sylvie Vilcoq	DANONE PRODUITS FRAIS FRANCE
Evan Bacchus	Costco Wholesale
Fabien Calleia	SIZEASE

Sven Dienelt	Hermann Hagemeyer GmbH & Co. KG
Marcel Ducceschi	Migros-Genossenschafts-Bund
Jonas Elander	Axfood Sverige AB
Max Engström	H&M
Plamen Iliev	DECATHLON
Jerome Lemay	DECATHLON
Dibyajeeban Mishra	DECATHLON
Sylvia Rubio Alegren	ICA Sverige AB
Hans Peter Scheidt	C & A SCS
Martijn Veerman	Customer Value
Joachim Wilkens	C & A SCS
Jeff Denton	AmerisourceBergen Corporation
Vladimir Dzalbo	Smartrac Technology Germany GmbH
Richard Fisher	DoD Logistics AIT Standards Office
Hajo Reissmann	Universitaetsklinikum Schleswig-Holstein
Hirokazu Nagai	Japan Pallet Rental Corporation
Thomas Burke	Institute of Food Technologists
Albert Arbones	GS1 Spain
Karen Arkesteyn	GS1 Belgium & Luxembourg
Andrea Arozamena	GS1 Mexico
Koji Asano	GS1 Japan
Andrea Ausili	GS1 Italy
Mahdi Barati	GS1 Iran
Xavier Barras	GS1 France
Jonas Batt	GS1 Switzerland
Arnaud Bonnefoy	GS1 France
Jonas Buskenfried	GS1 Sweden
Emanuela Casalini	GS1 Italy
Madalina Cernat	GS1 Romania
Anthony Chan	GS1 Hong Kong, China
Shawn Chen	GS1 Thailand
Luiz Costa	GS1 Brasil
Benjamin Couty	GS1 France
Amanda Creane	GS1 Ireland
Tim Daly	GS1 Ireland
Owen Dance	GS1 New Zealand
Michael Davis	GS1 Australia
Kevin Dean	GS1 Canada
huipeng deng	GS1 China
Sean Dennison	GS1 Ireland
Peta Ding	GS1 UK
Deniss Dobrovolskis	GS1 Sweden
Nipun Dogra	GS1 India
Xiaowen Dong	GS1 China
Gianluca Fazio	GS1 Argentina
Guilherme França	GS1 Brasil

Michele Francis Padayachee	GS1 South Africa
Jesper Kervin Franke	GS1 Denmark
Jean-Christophe Gilbert	GS1 France
Vanessa Giulieri	GS1 Italy
Alvin Goh	GS1 Singapore
Nicole Golestani	GS1 Canada
Heinz Graf	GS1 Switzerland
magali Granger	GS1 France
Marija Groznik Stankovic	GS1 Slovenia
János Gyuris	GS1 Hungary
Rami Habbal	GS1 UAE
Jason Hale	GS1 UK
Gary Hartley	GS1 New Zealand
Sandra Hohenecker	GS1 Germany
Hideki Ichihara	GS1 Japan
Yoshihiko Iwasaki	GS1 Japan
Yo Han Jeon	GS1 Korea
Yohan Jeon	GS1 Korea
Fiona (Zhitao) Jia	GS1 China
Sang Ik Jung	GS1 Korea
iliada karali	GS1 Association Greece
Kimmo Keravuori	GS1 Finland
Kazuna Kimura	GS1 Japan
Dora Kit	GS1 Hong Kong, China
Sabine Klaeser	GS1 Germany
Alexey Krotkov	GS1 Russia
Chris Lai	GS1 Hong Kong, China
Ildikó Lieber	GS1 Hungary
Xiaoyan Liu	GS1 China
Marisa Lu	GS1 Chinese Taipei
Ilka Machemer	GS1 Germany
Noriyuki Mama	GS1 Japan
Roberto Matsubayashi	GS1 Brasil
Riad Mechtari	GS1 Algeria
Terje Menkerud	GS1 Norway
Jan Merckx	GS1 Netherlands
Ephraim Mokheseng	GS1 South Africa
Adrien Molines	GS1 France
Naoko Mori	GS1 Japan
Daniel Mueller-Sauter	GS1 Switzerland
Prince Namane	GS1 South Africa
Jorge Andrés Nava Alanís	GS1 Mexico
Zubair Nazir	GS1 Canada
Daisuke Negishi	GS1 Japan
Alice Nguyen	GS1 Vietnam
Maciej Niemir	GS1 Poland

Staffan Olsson	GS1 Sweden
Manos Papadakis	GS1 Association Greece
Sebastián Perazzo	GS1 Argentina
Thiago Perez Rojas	GS1 Argentina
James Perng	GS1 Chinese Taipei
Bijoy Peter	GS1 India
Sarina Pielaat	GS1 Netherlands
Aruna Ravikumar	GS1 Australia
Paul Reid	GS1 UK
Zbigniew Rusinek	GS1 Poland
Nick Rusman	GS1 Netherlands
Sunny Sanam	GS1 Australia
Roxana Saravia Bulmini	GS1 Argentina
Yuki Sato	GS1 Japan
Sue Schmid	GS1 Australia
Eugen Sehorz	GS1 Austria
Pooja Sengupta	GS1 Australia
Xiaojing Shao	GS1 China
Yuko Shimizu	GS1 Japan
Marcel Sieira	GS1 Australia
Cesar Silvestre	GS1 Mexico
Olga Soboleva	GS1 Russia
Roko Staničić	GS1 Slovenia
Andrew Steele	GS1 Australia
Sylvia Stein	GS1 Netherlands
Jo Anna Stewart	GS1 US
Ralph Troeger	GS1 Germany
Frits van den Bos	GS1 Netherlands
Ricardo Verza Amaral Melo	GS1 Brasil
Linda Vezzani	GS1 Italy
Rocio Vizcarra	GS1 Argentina
Amber Walls	GS1 US
Yi Wang	GS1 China
Achim Wetter	GS1 Germany
Stephan Wijnker	GS1 Australia
Dirk Willekens	GS1 Belgium & Luxembourg
Connie Wong	GS1 Canada
Ruoyun Yan	GS1 China
Shawn Zhang	GS1 China
Victor Zhang	GS1 China
Marc Blanchet	Viagenie
SHREENIDHI BHARADWAJ	Syndigo
Scott Brown	1WorldSync, Inc.
Shawn Cady	Syndigo
Ed Collins	Brandbank
J.D. Kern	Syndigo

Sprague Ackley	Digimarc
Adnan Alattar	Digimarc
Philip Allgaier	bpcompass GmbH
Attilio Bellman	Antares Vision
Karim Ben Dakhlia	Dentsu Aegis Network
Jayson Berryhill	Envisible LLC
Dalibor Biscevic	Business Technologies Ltd
Megan Brewster	Impinj, Inc
Menno Bruil	H2Compute
Randy Burd	Kwikee, A Syndigo Company
Steffen Butschbacher	bpcompass GmbH
Tony Ceder	Charmingtrim
Robert Celeste	Center for Supply Chain Studies
Patrick Chanez	INEXTO SA
Grant Courtney	Be4ward ltd
Henk Dannenberg	NXP Semiconductors
Dilip Daswani	Qliktag Software (formally Zeebric LLC)
Cory Davis	Digimarc
Christophe Devins	Adents
Roland Donzelle	SQUARE / TINTAMAR
Chuck Evanhoe	Evanhoe & Associates, Inc.
Susan Flake	Zebra Technologies Corporation
Tomaz Frelih	Četrta pot,d.o.o.,Kranj
Mathieu Gallant	Optel Group
Ivan Gonzalez	recycl3R
Richard Graves	Phy
Danny Haak	Nedap
Steve Halliday	RAIN RFID Alliance
Mark Harrison	Milecastle Media Limited
Philip Heggelund	DuckScape Inc
John Herzig	Barcode Graphics Inc Canada
Bernie Hogan	Independent Consultant - Bernie Hogan
Dan James	Digimarc
Sandun Jayawardena	H2Compute
Margo Johnson	Transmute
Paul Kanwar	ScanTrust
Thomas Kühne	Goodstag GmbH
Sean Lockhead	Lockhead Consulting Group LLC
Andrew Love	Be4ward ltd
André Machado	TrustaTAG
Lee Metters	Domino Printing Sciences PLC
Joel Meyer	Digimarc
Mario Mira	Dentsu Aegis Network
Attila Sándor Nagy	infiCom.EU Co. Ltd.
ilteris oney	ecomes
Mitun Pandey	Goodstag GmbH

Tiphaine Paulhiac	Ambrosus Technologies
Fernando Pereira	Saphety Level SA
Justin Picard	ScanTrust
Scott Pugh	Jennason LLC
Tony Rodriguez	Digimarc
Octavio Rodriguez	Systech International
Zbigniew SAGAN	Advanced Track and Trace
Joannie Sauvageau	Optel Group
Kim Simonalle	Qliktag Software (formally Zeebric LLC)
Laurent TONNELIER	mobiLead
Andrew Verb	Bar Code Graphics, Inc.
Elizabeth Waldorf	TraceLink
Alex Winiarski	Winiarski Group
George Wright IV	Product Identification & Processing Systems
Shi Yu	Beijing REN JU ZHI HUI Technology Co. Ltd.
Pete Alvarez	GS1 Global Office
Phil Archer	GS1 Global Office
Lena Coulibaly	GS1 Global Office
Nadi (Scott) Gray	GS1 Global Office
Steven Keddie	GS1 Global Office
Neil Piper	GS1 Global Office
Craig Alan Repec	GS1 Global Office
Greg Rowe	GS1 Global Office

Log of Changes

Release	Date of Change	Changed By	Summary of Change
0.1	2020-10-20	Phil Archer	Separation of the compression standard separate from the original version 1.1. Setting up introductory material.
0.2	2020-10-27	Mark Harrison	Updates to support expression of EPC binary strings within compressed GS1 Digital Link URIs
0.3	2020-10-28	Mark Harrison	Corrected definition of `primaryIDcomponent` in section 3.7.1 to align with corrected definition in URI syntax chapter.
0.4	2020-11-11	Mark Harrison	Update and renumber flowcharts for compression and decompression. Update Table F.
0.5	2020-11-19	Phil Archer	Tidy up for commRev

Disclaimer

GS1®, under its IP Policy, seeks to avoid uncertainty regarding intellectual property claims by requiring the participants in the Work Group that developed this **GS1 Digital Link: Compression Standard** to agree to grant to GS1 members a royalty-free licence or a RAND licence to Necessary Claims, as that term is defined in the GS1 IP Policy. Furthermore, attention is drawn to the possibility that an implementation of one or more features of this Specification may be the subject of a patent or other intellectual property right that does not involve a Necessary Claim. Any such patent or other intellectual property right is not subject to the licencing obligations of GS1. Moreover, the agreement to grant licences provided under the GS1 IP Policy does not include IP rights and any claims of third parties who were not participants in the Work Group.

Accordingly, GS1 recommends that any organisation developing an implementation designed to be in conformance with this Specification should determine whether there are any patents that may encompass a specific implementation that the organisation is developing in compliance with the Specification and whether a licence under a patent or other intellectual property right is needed. Such a determination of a need for licencing should be made in view of the details of the specific system designed by the organisation in consultation with their own patent counsel.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR PARTICULAR PURPOSE, OR ANY WARRANTY OTHER WISE ARISING OUT OF THIS SPECIFICATION. GS1 disclaims all liability for any damages arising from use or misuse of this document, whether special, indirect, consequential, or compensatory damages, and including liability for infringement of any intellectual property rights, relating to use of information in or reliance upon this document.

GS1 retains the right to make changes to this document at any time, without notice. GS1 makes no warranty for the use of this document and assumes no responsibility for any errors which may appear in the document, nor does it make a commitment to update the information contained herein.

GS1 and the GS1 logo are registered trademarks of GS1 AISBL.

Table of Contents

1 Introduction

This section and its subsections are informative

GS1 defines a wide range of identifiers that underpin the supply chain and retail industry across the world. This document assumes the reader is familiar with these and the concept of GS1 Application Identifiers. If not, please see information on [GS1 identification Keys] and the [GENSPECS] for further background.

This work has been motivated by a number of trends. For example: the desire among retailers to move to 2D barcodes that can carry more information than just the GTIN; the problems of multiple barcodes causing scanning errors through conflicts which suggests a need for a single but multipurpose barcode; the growing expectation among consumers that more information is available online about the products they're considering buying; the brand owner concept of the pack as a media channel linking to multimedia experiences, and more.

As a result of GS1 Digital Link, it is possible to represent GS1 identification keys consistently within Web addresses as well as within barcodes containing Web addresses, such that a single identification approach can support both product identification for supply chain applications *and* a link to online material for consumer and business partner interactions. It's this dual functionality and enormous flexibility that is currently not possible when, for example, Brand Owners embed an unstructured Web page address in a QR Code®[1].

The scope of the work accommodates all Class 1 and Class 2 GS1 Keys and Key qualifiers (e.g., serial number, batch number, consumer product variant) and other relevant attributes as the same technologies are equally applicable to SSCCs, GLNs, GIAIs, GRAIs, GSRNs etc. While the syntax can support Class 2 Keys, it is up to the Class 2 Issuing Agencies to determine whether it's fit for their use. For Class 3 GS1 Keys, GS1 welcomes bilateral discussions with Issuing Agencies to see where alignment is possible.

This GS1 standard references a number of third-party standards from the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C).

1.1 How the GS1 Digital Link standard documents fit together

Rather than one very long document containing every detail, as of version 1.2, the GS1 Digital Link standard comprises 4 discrete documents:

URI syntax

This document provides some of the background to the design of GS1 Digital Link, highlighting existing techniques and practices that underpin the World Wide Web, and applying those to the GS1 system. The normative portions set out the detailed syntax of Web addresses (HTTP URIs) that encode GS1 identifiers with exactly the same precision and expressivity as the AI-based element syntax used across the GS1 system, notably in the GS1 General Specifications. The GS1 Digital Link URI syntax distinguishes between primary keys, such as GTIN and GLN, key qualifiers, such as batch/lot and GLN extension, and attributes such as expiry date and ship to address. The GS1 Digital Link URI syntax is the foundation on which all other aspects of the standard are built.

Compression (this document)

A GS1 Digital Link URI that contains a set of identifiers and attributes may exceed the capacity of some data carriers. This document defines a compression/decompression algorithm that minimises the length of those Web URIs while retaining two critical features: 1) that the compressed form is still a URL on the same domain as the uncompressed form, that is, there is no change in ownership of the URL; 2) that it can be decompressed and the GS1 keys extracted *without* an online lookup.

Resolution

A GS1 Digital Link URI is a particular form of URL and *can* be used in exactly same same way as any other URL (this is an important design feature). However, it can also be the gateway to multiple sources of information, both human and machine-readable. This document defines how the keys in

[1] Unless otherwise specified, the term 'QR Code®' refers to the widely used *ISO/IEC 18004 QR Code*®, excluding the GS1 QR Code that recognises the FNC1 character. 'QR Code' is a registered trademark of Denso Wave, a subsidiary of Denso Corporation. Both the *ISO/IEC 18004 QR Code*® and GS1 QR Code follow the encoding scheme described in ISO/IEC 18004 Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification, 3rd edition 2015-02-01.

a GS1 Digital Link URI can be 'resolved' to those information sources in such a way that information systems and apps can discover them automatically. Resolvers are what makes the standard operational for the GS1 community and the industries served.

Semantics

Devices like scanners and point of sale terminals, PIM systems, product catalogues and more that are designed specifically to work with GS1 identifiers and data carriers, are all programmed to function within that particular framework. GS1 Digital Link puts things like GTINs, SSCCs and GRAIs onto the Web alongside countless other identifiers and ways of working. This document expresses the meaning behind the GS1 Digital Link standard in a way that the Web at large can understand and process. It makes use of, and extends, the GS1 Web Vocabulary.

1.2 Typographical conventions used in this document

This document includes a lot of examples of GS1 Digital Link URIs such as:

```
https://example.com/gtin/{gtin} and
https://example.org/414/{gln}/254/{glnExtension}
https://example.org/01/{gtin}{?exp}
```

The use of the monospace font indicates that the text has meaning for computers. Further, these examples follow the convention used in [RFC 6570]. The places where the values of variables should be inserted are written in braces, so, for example, {gtin} means "insert gtin here". All other text in the URI is a literal string to be used as written. As explained in [RFC 2606] and [RFC 6761], the domains example.com, example.org and example.net are second-level domain names reserved by the Internet Assigned Numbers Authority (IANA) for use in documentation. These should be understood as a placeholder for any registered second-level domain name.

2 Conformance to GS1 Digital Link

This section is normative

The GS1 Digital Link standard comprises a number of discrete documents against which conformance can be asserted. There is no single conformance statement for the entirety of GS1 Digital Link. It is therefore inappropriate to make a formal claim of broad conformance without citing the specific standard with which conformance is claimed.

It is worth noting that, whether compressed or not, a GS1 Digital Link URI, like any Web URI or URL, does not have any intrinsic meaning. It may be treated in exactly the same way as any URL. It is only if it is parsed by a GS1-aware system that GS1 application identifiers and their values can be extracted and processed. Examples of such systems include scanners that may treat a GS1 Digital Link URI as an alternative syntax to element strings, and conformant GS1 resolvers. Applications SHALL NOT assume that a URL that follows the syntax defined in this standard will point to a resolver. One way to test whether a Web URI does or does not point to a GS1 conformant resolver is to check for the presence of a Resolver Description File in the relevant Well-Known location /.well-known/gs1resolver [RFC 8615]. Details of the Resolver Description File are defined in GS1 DigitalLink: Resolution [DL-Resolution].

3 Compression and decompression

This section and all subsections, except subsections 3.1 – 3.3, are normative.

3.1 Purpose of compression

GS1 Digital Link URIs can be used with any data carrier that can directly embed an entire URL. These include NFC tags, regular ISO 18004 QR codes, digital watermarks, Data Matrices, etc. Compression may be useful when encoding a GS1 Digital Link URI within a data carrier that has limited memory capacity.

Compression as defined in this document results in a compressed GS1 Digital Link URI that is shorter in length than the original. For optical data carriers, this should reduce the total count of modules, which in turn leads to improvements in read and print reliability, particularly when the physical footprint of the data carrier may be constrained.

Version 1.2 of the GS1 Digital Link standard expands the core compression algorithm to add support for recently introduced GS1 Application Identifiers, primarily those for GS1 Scan4Transport. A further new feature is support for expression of EPC binary strings, although their decoding should be in accordance with the GS1 Tag Data Standard [TDS] and may make use of the machine-readable artifacts and framework of the GS1 Tag Data Translation standard [TDT]. Three compression headers are now used to support decoding of GTIN-8, GTIN-12 and GTIN-13 values, preserving their length, instead of requiring every GTIN value to be padded to 14 digits on compression and decompression.

3.2 Technical requirements extracted from the business requirements

Business Requirements Analysis identified the need for:

- Reversible encoding such that GS1 identifiers can be extracted without an online lookup.
- Flexibility, potentially only affecting part of the URI, including the option to keep the primary identification key (e.g. GTIN) uncompressed.
- That the compressed GS1 Digital Link URI is still a valid URL that requires no processing before making a network request.

If the output of the compression algorithm still exceeds the capacity of the data carrier then an even shorter, unstructured URL may point to a class 2 resolver (see [DL-Resolution]) from which a conformant GS1 Digital Link may be obtained.

3.3 Viability of types of compression techniques and opportunities for compression

The need for reversible encoding (i.e. the ability to decompress what was compressed) means that it is only realistic to consider lossless compression techniques.

This eliminates lossy compression techniques such as those used in compression of JPEG images.

Typical values for GS1 Application Identifiers do not usually have several contiguous repeated characters, so run-length compression techniques are also inappropriate.

There is a variety of possible formats for the values of GS1 Application Identifiers. These include:

- Fixed-length all-numeric strings
- Variable-length all-numeric strings
- Variable-length alphanumeric strings
- Fixed-length all-numeric strings followed by a variable-length alphanumeric component

At the lowest level, computer systems use binary encoding to represent information, data structures and character sequences.

Using a full byte (8 bits) to represent a numeric digit uses more bits than the minimum needed; 4 bits are capable of encoding numeric digits 0-9 and hexadecimal characters a-f; approximately 3.32

bits per digit are required to encode a numeric string as a binary integer. This therefore represents a significant opportunity for compression particularly for all-numeric strings and dates.

When data is encoded efficiently in binary format, this is close to the mathematical limit of the most compact format that supports lossless compression and decompression.

3.4 Structure of the compressed string

3.4.1 Conversion between binary and URI-safe base 64 alphabet

A binary string may be an efficient way of encoding data but a compressed GS1 Digital Link URI needs to convert this into a compact string of characters. The standard for Base16, Base32, and Base64 Data Encodings [RFC4648] provides details of a URI-safe base 64 character set that uses 6 bits to encode each character, using the following code table and a padding character of "=" :

Index	Char	Index	Char	Index	Char	Index	Char
0 000000	A	**16** 010000	Q	**32** 100000	g	**48** 110000	w
1 000001	B	**17** 010001	R	**33** 100001	h	**49** 110001	x
2 000010	C	**18** 010010	S	**34** 100010	i	**50** 110010	y
3 000011	D	**19** 010011	T	**35** 100011	j	**51** 110011	z
4 000100	E	**20** 010100	U	**36** 100100	k	**52** 110100	0
5 000101	F	**21** 010101	V	**37** 100101	l	**53** 110101	1
6 000110	G	**22** 010110	W	**38** 100110	m	**54** 110110	2
7 000111	H	**23** 010111	X	**39** 100111	n	**55** 110111	3
8 001000	I	**24** 011000	Y	**40** 101000	o	**56** 111000	4
9 001001	J	**25** 011001	Z	**41** 101001	p	**57** 111001	5
10 001010	K	**26** 011010	a	**42** 101010	q	**58** 111010	6
11 001011	L	**27** 011011	b	**43** 101011	r	**59** 111011	7
12 001100	M	**28** 011100	c	**44** 101100	s	**60** 111100	8
13 001101	N	**29** 011101	d	**45** 101101	t	**61** 111101	9
14 001110	O	**30** 011110	e	**46** 101110	u	**62** 111110	-
15 001111	P	**31** 011111	f	**47** 101111	v	**63** 111111	_

Such a URI-safe base 64 alphabet can be used to represent the binary string representation as characters A-Z a-z 0-9 hyphen and underscore, without requiring any of these characters to be percent encoded within a URI.

3.4.2 Various defined tables for GS1 AIs (length, format)

The GS1 General Specifications includes a table (appearing as Figure 3.2-1 of GS1 General Specifications v19 [GENSPECS]), which includes a format for each defined GS1 Application Identifier. The information in that table has been used to construct Table F shown below.

Table F indicates the expected format for the value of each GS1 Application Identifier.

	First Component			Second Component		
AI	**Encoding E** N = numeric, X=alphanumeric	**Fixed Length L**	**Max. Length M**	**Encoding E** N = numeric, X=alphanumeric	**Fixed Length L**	**Max. Length M**
00	N	18				
01	N	14				
02	N	14				
10	X		20			
11	N	6				
12	N	6				
13	N	6				
15	N	6				
16	N	6				
17	N	6				
20	N	2				
21	X		20			
22	X		20			
240	X		30			
241	X		30			
242	N		6			
243	X		20			
250	X		30			
251	X		30			
253	N	13		X		17
254	X		20			
255	N	13		N		12
30	N		8			

3100-3105 3110-3115 3120-3125 3130-3135 3140-3145 3150-3155 3160-3165	N	6				
3200-3205 3210-3215 3220-3225 3230-3235 3240-3245 3250-3255 3260-3265 3270-3275 3280-3285 3290-3295	N	6				
3300-3305 3310-3315 3320-3325 3330-3335 3340-3345 3350-3355 3360-3365 3370-3375	N	6				
3400-3405 3410-3415 3420-3425 3430-3435 3440-3445 3450-3455 3460-3465 3470-3475 3480-3485 3490-3495	N	6				
3500-3505 3510-3515 3520-3525 3530-3535 3540-3545 3550-3555 3560-3565 3570-3575	N	6				
3600-3605 3610-3615 3620-3625 3630-3635 3640-3645 3650-3655 3660-3665 3670-3675 3680-3685 3690-3695	N	6				
37	N		8			
3900-3909	N		15			
3910-3919	N	3		N		15
3920-3929	N		15			

3930-3939	N	3		N		15
3940-3943	N	4				
3950-3953	N	6				
400	X		30			
401	X		30			
402	N	17				
403	X		30			
410 - 417	N	13				
420	X		20			
421	N	3		X		9
422	N	3				
423	N	3		N		12
424	N	3				
425	N	3		N		12
426	N	3				
427	X		3			
4300	X		35			
4301	X		35			
4302	X		70			
4303	X		70			
4304	X		70			
4305	X		70			
4306	X		70			
4307	X	2				
4308	X		30			
4310	X		35			
4311	X		35			
4312	X		70			
4313	X		70			
4314	X		70			
4315	X		70			
4316	X		70			

4317	X	2				
4318	X		20			
4319	X		30			
4320	X		35			
4321	N	1				
4322	N	1				
4323	N	1				
4324	N	10				
4325	N	10				
4326	N	6				
7001	N	13				
7002	X		30			
7003	N	10				
7004	N		4			
7005	X		12			
7006	N	6				
7007	N	6		N		6
7008	X		3			
7009	X		10			
7010	X		2			
7020	X		20			
7021	X		20			
7022	X		20			
7023	X		30			
7030-7039	N	3		X		27
710 - 714	X		20			
7230-7239	X		30			
8001	N	14				
8002	X		20			
8003	N	14		X		16
8004	X		30			
8005	N	6				

8006	N	18				
8007	X		24			
8008	N	8		N		4
8009	X		50			
8010	X		30			
8011	N		12			
8012	X		20			
8013	X		30			
8017	N	18				
8018	N	18				
8019	N		10			
8020	X		25			
8026	N	18				
8110	X		70			
8111	N	4				
8112	X		70			
8200	X		70			
90	X		30			
91-99	X		90			

3.4.3 How each GS1 AI key : value pair is encoded in binary

The binary string consists of a concatenation of binary string encodings for each encoded key=value pair. For GS1 Application Identifiers, the key is usually the numeric GS1 Application Identifier. However, Table Opt (section 3.5) of the compression algorithm also defines 2-character hexadecimal Optimisation Codes that correspond to a pre-defined sequence of one or more GS1 Application Identifiers.

Each binary string encoding begins with 8 bits, which are interpreted as two hexadecimal character in the range 0-9 a-f.

If both of those characters are in the range 0-9, then they are interpreted as the first two digits of a GS1 Application Identifier key.

All current GS1 Application Identifiers consist of 2-4 digits and it is possible to use the first two digits to determine whether the GS1 Application Identifier consists of 2, 3 or 4 digits. These rules are summarised in Table P below.

Table P indicates for any initial two digits, what is the total length of the numeric AI key

First 2 digits	*AI key Length*	*First 2 digits*	*AI key Length*	*First 2 digits*	*AI key Length*	*First 2 digits*	*AI key Length*	*First 2 digits*	*AI key Length*
00	2	**20**	2	**34**	4	**72**	4	**96**	2
01	2	**21**	2	**35**	4	**71**	3	**95**	2
02	2	**22**	2	**36**	4	**80**	4	**97**	2
10	2	**23**	3	**37**	2	**81**	4	**98**	2
11	2	**24**	3	**39**	4	**82**	4	**99**	2
12	2	**25**	3	**40**	3	**90**	2		
13	2	**30**	2	**41**	3	**91**	2		
15	2	**31**	4	**42**	3	**92**	2		
16	2	**32**	4	**43**	4	**93**	2		
17	2	**33**	4	**70**	4	**94**	2		

3.4.4 Length indicators and Encoding indicators

Length indicators are used to support more efficient encoding of values where the value is permitted to be variable-length. No length indicator is used if the value is defined to be a fixed-length field.

Encoding indicators are used to support more efficient encoding of values where the value is permitted to be alphanumeric (including specified permitted symbol characters). No encoding indicator is used if the value is defined to be a numeric field.

	Encoding indicator used?	Length indicator used?
Fixed-length all-numeric strings	NO	NO
Variable-length all-numeric strings	NO	YES
Variable-length alphanumeric string	YES	YES
Fixed-length all-numeric strings followed by a variable-length alphanumeric component	YES - with second component	YES - with second component

3.4.4.1 Encoding indicator

A 3-bit encoding indicator is needed wherever alphanumeric or symbol characters are permitted for a value or within a value component. No encoding indicator is used in situations where the value is defined to be all-numeric (e.g. such as the value of a GTIN or SSCC).

Encoding value (decimal)	Encoding value (binary)	Character encoding
0	0 0 0	All-numeric string encoded as integer at ≈ 3.32 bits per character
1	0 0 1	Lower-case hexadecimal characters at 4 bits per character
2	0 1 0	Upper-case hexadecimal characters at 4 bits per character
3	0 1 1	URI-safe base64 characters A-Z a-z 0-9 hyphen and underscore at 6 bits per character
4	1 0 0	ASCII characters in range 0-127 at 7 bits per character
5	1 0 1	*reserved for other encoding (to be defined in future)*
6	1 1 0	*reserved for other encoding (to be defined in future)*
7	1 1 1	**First Extension point to a longer encoding indicator (e.g. 6 bits) providing a further 7 values plus Second Extension point.**

3.4.4.2 Length indicator

The GS1 General Specifications define that a number of GS1 Application Identifiers have values or value components whose length is variable, up to a maximum permitted length. For example, AI (10) for Batch/Lot and AI (21) for Serial Number both permit a variable-length alphanumeric value up to 20 characters.

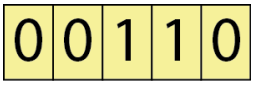
A length indicator is used where the length of a value or value component is not of predefined length.

The length indicator consists of a number of bits, n, whose binary value corresponds to L where L is the actual length of the value of a variable-length value or value component. The number of bits (n) is selected depending of the maximum permitted length (Lmax) for the value or value component. A length indicator of n bits permits expression of a length in the range 0 to (2n − 1).

For convenience, the following table provides values for n, the number of bits used for the length indicator.

Length range	n	Some examples of GS1 AIs that use this range	Examples of variable-length notation used in GS1 General Specifications Figure 3.2-1
0-3	2	7010	N..2
0-7	3	242	N..6
0-15	4	424	N..12
0-31	5	10, 21, 22	X..20
		240, 241,250,251	X..30
0-63	6	8007 8009	X..34 X..50
0-127	7	8110 91 - 99	X..70 X..90

The following example shows a 5-bit length indicator (n=5) in which the binary value is 00110 (equivalent to value 6 in base 10 / decimal). This example might be used to indicate a batch/lot or serial number whose actual value is 6 characters in length.



GS1 Application Identifiers 10, 21 and 22 would each use a 5-bit length indicator because they permit values up to 20 characters in length, so a 4-bit length indicator is insufficient, since it would only permit lengths in the range 0-15, so it is necessary to select a 5-bit length indicator instead, since its range 0-31 can accommodate 0-20.

For a GS1 Application Identifier whose value is defined as variable-length up to L_{max} characters or digits, the value encoded within the length indicator SHALL NOT exceed L_{max} permitted for that GS1 Application Identifier, even if a larger integer value could be expressed within the n bits of the length indicator.

Explanation: Although an n-bit length indicator mathematically supports lengths in the range 0 - $(2^n - 1)$, the maximum length permitted for that GS1 Application Identifier (see Figure 3.2-1 in the GS1 General Specifications) takes precedence.

During compression or decompression, the implementation of an algorithm SHALL check that the value encoded within the length indicator does not exceed the corresponding maximum length permitted by the GS1 General Specifications, reflected in Table F.

For variable-length numeric values, the length indicator is immediately followed by the binary-encoded value.

For variable-length alphanumeric values, the encoding indicator (always 3 bits in this version of the compression algorithm) SHALL always be followed by the length indicator. This SHALL be followed by the binary-encoded value unless the actual value is a zero-length string.

The following table indicates the number of bits n_v that should be read for the value.

Encoding -->	000	001 or 010	011	100
Length indicator n	n_v= ceiling(n * log(10)/log(2))	$n_v = 4n$	$n_v = 6n$	$n_v = 7n$

3.5 Optimised encoding of combinations of GS1 Application Identifiers

Capacity exists to support further compression of combinations of GS1 Application Identifiers. The table below lists a number of pre-defined sequence of GS1 Application Identifiers. Instead of encoding each numeric GS1 Application Identifier key using 4 bits per digit, a single 8 bit code indicates the pre-defined sequence. The table also includes some values reserved for future use as well as two values dedicated for expressing an EPC binary string, explained in further detail in section 3.5.1.

Table Opt

Code	Sequence of GS1 Application Identifiers	Meaning	Usage
0A	["01","22"]	GTIN + consumer product variant	retail, CPG metrics
0B	["01","10"]	GTIN + batch/lot	retail, recalls
0C	["01","21"]	GTIN + serial	Serialisation
0D	["01","17"]	GTIN + expiry date	retail, fresh food
0E	["01","7003"]	GTIN + expiry date&time	retail, fresh food
0F	["01","30"]	GTIN + count	variable measure
1A	["01","10","21","17"]	GTIN + batch/lot+serial+expiry	Pharma
1B	["01","15"]	GTIN + best before	retail, fresh food
1C	["01","11"]	GTIN + production date	retail, food traceability
1D	["01","16"]	GTIN + sell by date	retail, fresh food
1E	["01","91"]	GTIN + company internal information	brand protection
1F	["01","10","15"]	GTIN + batch/lot + best before	retail, fresh food
2A	["01","3100"]	GTIN + weight (kg)	variable measure
2B	["01","3101"]	GTIN + weight (kg)	variable measure
2C	["01","3102"]	GTIN + weight (kg)	variable measure
2D	["01","3103"]	GTIN + weight (kg)	variable measure
2E	["01","3104"]	GTIN + weight (kg)	variable measure
2F	["01","3105"]	GTIN + weight (kg)	variable measure
3A	["01","3200"]	GTIN + weight (lbs)	variable measure
3B	["01","3201"]	GTIN + weight (lbs)	variable measure
3C	["01","3202"]	GTIN + weight (lbs)	variable measure
3D	["01","3203"]	GTIN + weight (lbs)	variable measure

3E	["01","3204"]	GTIN + weight (lbs)	variable measure
3F	["01","3205"]	GTIN + weight (lbs)	variable measure
9A	["8010","8011"]	CPID + CPID serial number	Automotive
9B	["8017","8019"]	GSRN-P + SRIN	service relationships (provider)
9C	["8018","8019"]	GSRN + SRIN	service relationships (recipient)
9D	["414","254"]	physical location GLN + GLN extension	Locations
A0	["01","3920"]	GTIN + amount payable	variable measure
A1	["01","3921"]	GTIN + amount payable	variable measure
A2	["01","3922"]	GTIN + amount payable	variable measure
A3	["01","3923"]	GTIN + amount payable	variable measure
A4	["01","3924"]	GTIN + amount payable	variable measure
A5	["01","3925"]	GTIN + amount payable	variable measure
A6	["01","3926"]	GTIN + amount payable	variable measure
A7	["01","3927"]	GTIN + amount payable	variable measure
A8	["01","3928"]	GTIN + amount payable	variable measure
A9	["01","3929"]	GTIN + amount payable	variable measure
C0	["255","3900"]	GCN + amount payable	Coupons
C1	["255","3901"]	GCN + amount payable	Coupons
C2	["255","3902"]	GCN + amount payable	Coupons
C3	["255","3903"]	GCN + amount payable	Coupons
C4	["255","3904"]	GCN + amount payable	Coupons
C5	["255","3905"]	GCN + amount payable	Coupons
C6	["255","3906"]	GCN + amount payable	Coupons
C7	["255","3907"]	GCN + amount payable	Coupons
C8	["255","3908"]	GCN + amount payable	Coupons
C9	["255","3909"]	GCN + amount payable	Coupons
CA	["255","3940"]	GCN + percentage discount	Coupons
CB	["255","3941"]	GCN + percentage discount	Coupons
CC	["255","3942"]	GCN + percentage discount	Coupons

CD	["255","3943"]	GCN + percentage discount	Coupons
7A	EPC binary string – option A (expressed as hexadecimal encoding)		
7B	EPC binary string – option B (expressed as file-safe base 64 encoding)		
7C	Reserved for future capacity (longer compression headers)		
7D	Reserved for future capacity (longer compression headers)		
7E	Reserved for future capacity (longer compression headers)		
7F	Reserved for future capacity (longer compression headers)		
4A – 4F	*Reserved*	*6 values reserved for future use*	
50 – 5F	*Reserved*	*16 values reserved for future use*	
60 – 6F	*Reserved*	*16 values reserved for future use*	
70 – 79	*Reserved*	*10 values reserved for future use*	
80 – 8F	*Reserved*	*16 values reserved for future use*	
9E, 9F	*Reserved*	*2 values reserved for future use*	
AA	["01"] GTIN-8 encoded using integer encoding in 27 bits – see section 3.5.3		
AB	["01"] GTIN-12 encoded using integer encoding in 40 bits – see section 3.5.3		
AC	["01"] GTIN-13 encoded using integer encoding in 44 bits – see section 3.5.3		
AD-AF	*Reserved*	*3 values reserved for future use*	
B0 – BE	*Reserved*	*15 values reserved for future use*	
BF	*Compact Boolean for (4321),(4322),(4323)* *Special handling described in flowcharts C9 and D17 and section 3.5.2* *A 6-bit field follows, using 2 bits per Boolean value*		
CE,CF	*Reserved*	*Reserved for future use*	
D0 – DF	*Reserved*	*Reserved for expressing version number of compression algorithm*	
E0 – EF	*Reserved*	*16 values reserved for non-collision with non-standard compression algorithms*	Means that no GS1 standard compressed string SHALL begin with a 4-7
F0 – FF	*Reserved*	*reserved for indicating non-GS1 key:value pairs within compressed string*	
		= 112 values reserved for future use	

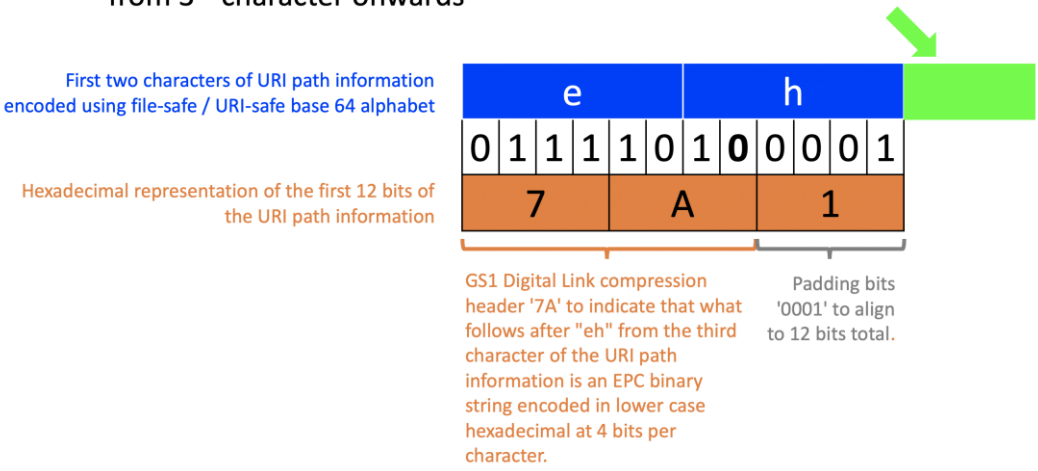
3.5.1 Expression of EPC binary string within a compressed GS1 Digital Link URI

Version 1.2 of the GS1 Digital Link standard introduces the capability to express the binary string encoding of an Electronic Product Code (EPC) identifier within a compressed GS1 Digital Link URI by using the special compression headers '7A' and '7B' to signal this.

As shown in the figures below, hexadecimal compression headers '7A' and '7B' correspond to the binary strings 01111010 and 01111011 respectively. When these are appended with binary string 0001, the resulting 12 bit binary strings 011110100001 and 011110110001 appear in the file-safe / URI-safe base 64 alphabet as the characters 'eh' and 'ex' respectively.
These are used to support two alternative options for expressing an EPC binary string.

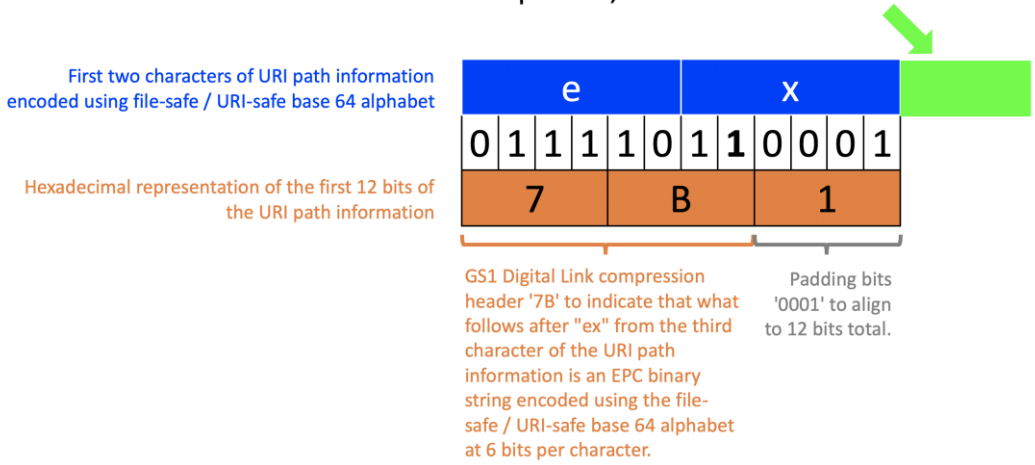
Option A:

URI mnemonic:
'eh' → EPC binary string in lower-case hexadecimal from 3rd character onwards



Option B:

URI mnemonic:
'ex' → EPC binary string in alphanumeric ('X') using an eXtended alphabet, the file-safe URI-safe base 64 alphabet, from 3rd character onwards



3.5.1.1 Option A: Hexadecimal encoding using compression header '7A' / string prefix 'eh'

If the compressed string of a GS1 Digital Link URI begins with '**eh**' then the characters following '**eh**' should be lower-case hexadecimal characters that correspond to the binary encoding of an EPC identifier, one lower-case hexadecimal character for each group of 4 bits. The EPC binary string begins with the EPC header as defined within the GS1 Tag Data Standard [TDS].

To construct a compressed GS1 Digital Link URI, convert an EPC binary string to lower-case hexadecimal then append to 'eh' to form the compressed string. Append the compressed string to a URI stem that excludes a query string or anchor fragment, ensuring that there is exactly one forward slash preceding it.

Worked example:

https://example.com/**eh**30164596f40c0e5cbe991a83

Compression header '**7A**' and compression string prefix '**eh**' indicate that what follows after '**eh**' is a hexadecimal encoding of an EPC binary string in which each hexadecimal character corresponds to 4 bits.

3	0	1	6	4	5	9	6	f	4	0	c	0	e	5	c	b	e	9	9	1	a	8	3
0011	0000	0001	0110	0100	0101	1001	0110	1111	0100	0000	1100	0000	1110	0101	1100	1011	1110	1001	1001	0001	1010	1000	0011

The first 8 bits, '00110000' are the EPC header for the SGTIN-96 EPC scheme.

Using the details defined in the GS1 Tag Data Standard, this binary string can be decoded to

urn:epc:id:sgtin:9528765.012345.123456789123

which is equivalent to element string **(01)**09528765123457**(21)**123456789123

which is equivalent to https://example.com**/01/**09528765123457**/21/**123456789123

3.5.1.2 Option B: File-safe base 64 encoding using compression header '7B' / string prefix 'ex'

If the compressed string of a GS1 Digital Link URI begins with '**ex**' then the characters following '**ex**' should be characters from the file-safe URI-safe base 64 alphabet (see section 3.4.1) that correspond to the binary encoding of an EPC identifier, using one such character for each successive group of 6 bits. The EPC binary string begins with the EPC header as defined within the GS1 Tag Data Standard.

To construct a compressed GS1 Digital Link URI, convert an EPC binary string to file-safe / URI-safe base64 characters at 6 bits per character then append to 'eh' to form the compressed string. Append the compressed string to a URI stem that excludes a query string or anchor fragment, ensuring that there is exactly one forward slash preceding it.

Worked example:

https://example.com/**ex**MBZFlvQMDly-mRqD

Compression header '7B' and compression string prefix '**ex**' indicates that what follows after '**ex**' is an encoding of an EPC binary string using the file-safe / URI-safe base64 alphabet (see Figure 3.4.1 and RFC 4648), in which each character corresponds to 6 bits.

M	B	Z	F	l	v	Q	M	D	l	y	-	m	R	q	D
001100	000001	011001	000101	100101	101111	010000	001100	000011	100101	110010	111110	100110	010001	101010	000011

The first 8 bits, '00110000' are the EPC header for the SGTIN-96 EPC scheme.

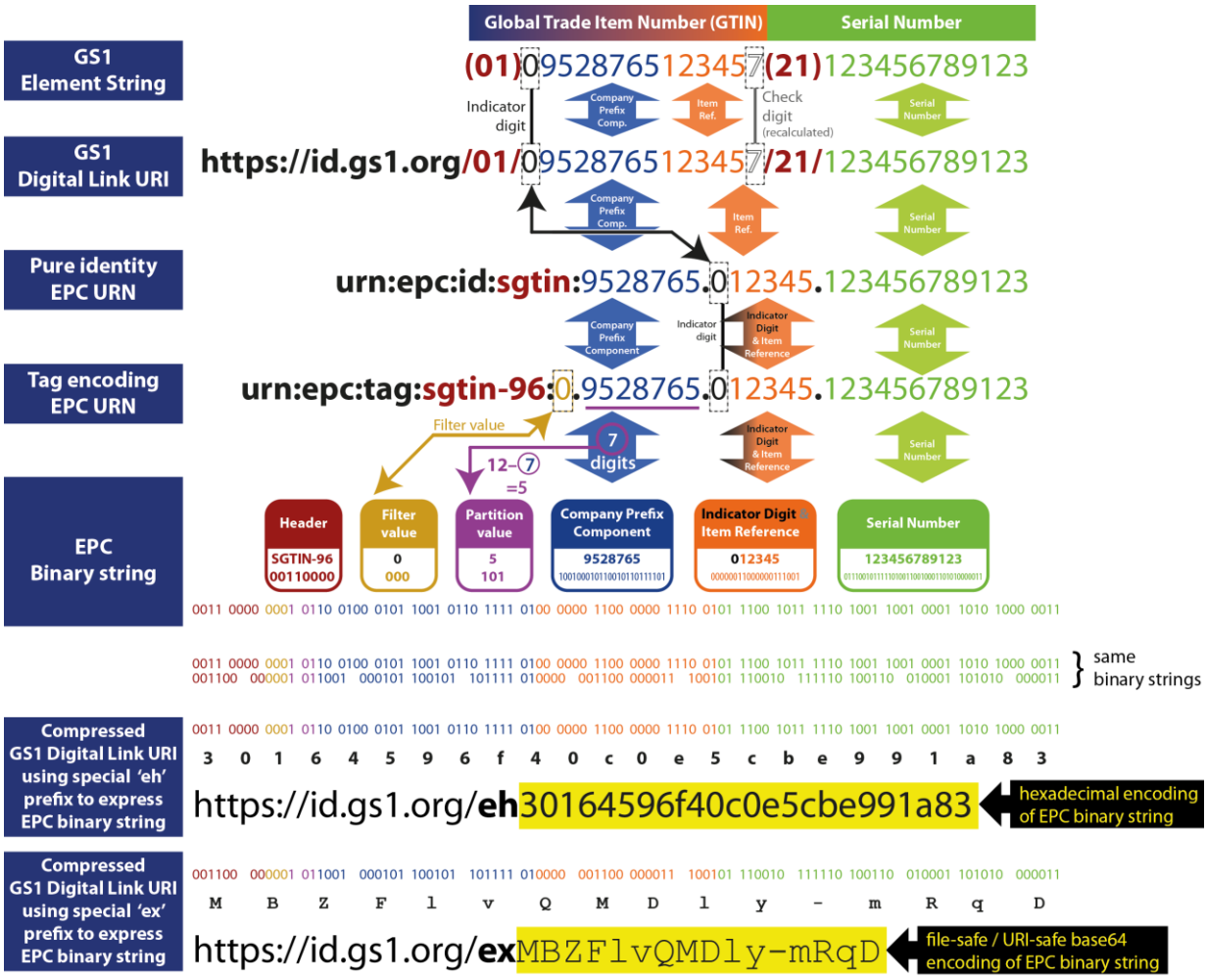
Using the details defined in the GS1 Tag Data Standard, this binary string can be decoded to

urn:epc:id:sgtin:9528765.012345.123456789123

which is equivalent to element string **(01)**09528765123457**(21)**123456789123

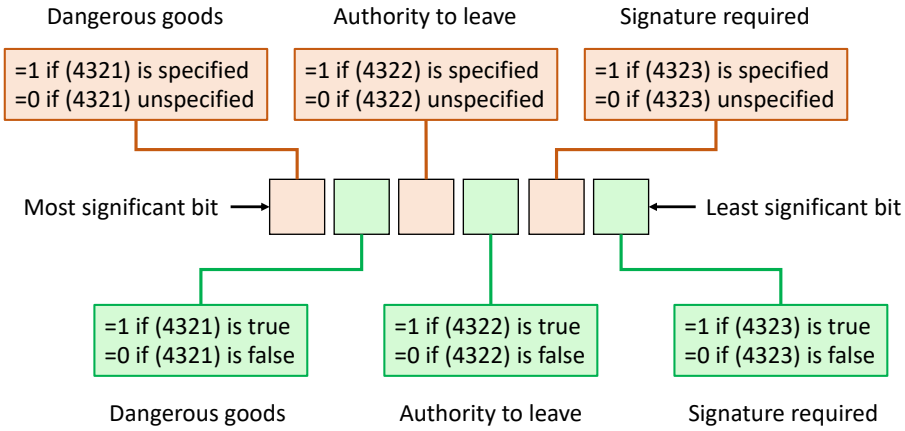
which is equivalent to https://example.com**/01/**09528765123457**/21/**123456789123

The figure below shows a graphical summary of the worked examples for option A (using special prefix 'eh') and option B (using special prefix 'ex') to express the same EPC binary string within a compressed GS1 Digital Link URI. The GS1 Tag Data Standard defines how to decode an EPC binary string to the EPC URN formats or to GS1 element strings.

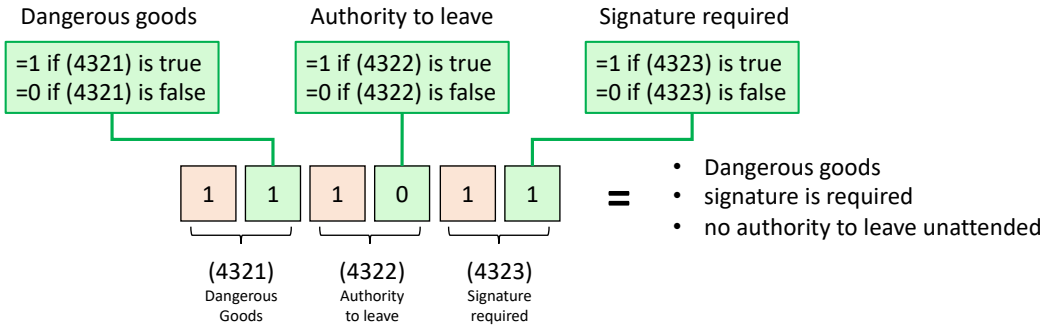


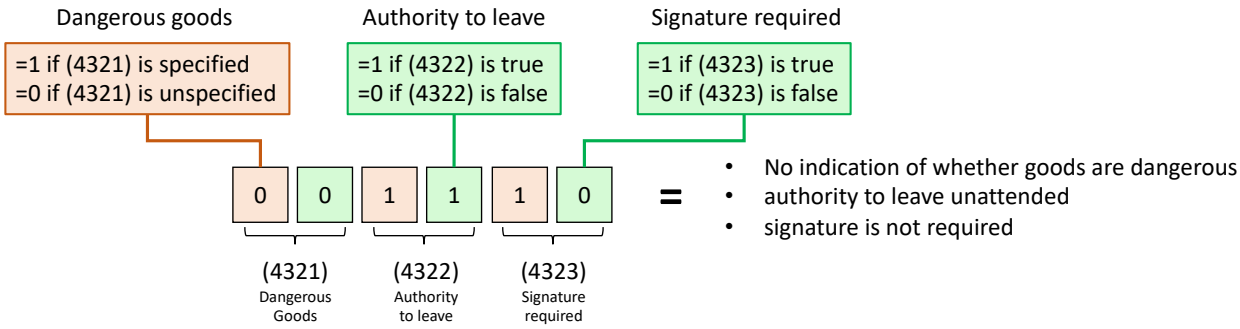
3.5.2 Compact encoding of Boolean values for GS1 Application Identifiers (4321)-(4323)

Compression header "BF" is reserved for compact encoding of a combination of the values of three GS1 Application Identifiers (4321), (4322) and (4323) within a 6-bit value that immediately follows compression header "BF", as shown in the diagram below.



The following diagram provides two examples to show how to encode or decode the 6-bit field that follows special compression header "BF"





3.5.3 Preservation of length when compressing GTIN-8, GTIN-12 and GTIN-13

In GS1 element strings, the GTIN is always expressed as a 14 digit field. For GTIN-8, GTIN-12 and GTIN-13 this is achieved by padding at the left with 6, 2 or 1 digits of '0'.

In general, GS1 Digital Link URIs do not require a GTIN value to be padded to 14 digits except in the canonical GS1 Digital Link URI representation.

The following compression headers are defined in v1.2 to express GTIN-8, GTIN-12 and GTIN-13 respectively:

Compression header or 8-bit Application Identifier	Meaning	Read/write next N bits after Compression header or 8-bit Application Identifier. (Pad at left with '0' if necessary to reach the specified bit length)	Interpret as M-digit GTIN (Pad at left with '0' if necessary to reach the specified number of digits)
01	GTIN-14	47 bits	14 digits
AC	GTIN-13	44 bits	13 digits
AB	GTIN-12	40 bits	12 digits
AA	GTIN-8	27 bits	8 digits

3.6 Examples of binary encoding of GS1 Application Identifiers

This section is informative

This section provides some worked examples to illustrate how GS1 Application Identifiers and their values can be efficiently compressed in binary strings.

3.6.1 GS1 Application Identifiers whose values are all-numeric and fixed-length

The following two examples show binary encoding of GS1 Application Identifiers where the value is both all-numeric and fixed length. In this situation, the GS1 Application Identifier (encoded at four bits per digit) is immediately followed by the value, encoded as a binary string corresponding to an integer value. The number of bits N required for an all-numeric fixed length string of D digits is given by the formula:

$$N = \text{ceiling}(D * \log(10)/\log(2))$$

The binary value is left-padded (highest significant bits set to ‘0’) in order to reach the total of N bits.

3.6.1.1 Binary encoding example for GTIN: (01)10614141123459

<table>
<tr><td colspan="8">GS1 AI (01)</td><td colspan="47">Value of AI (01) is fixed-length numeric
and left-padded to a total of 47 bits = ceiling (14*log(10)/log(2))</td></tr>
<tr><td colspan="4">0</td><td colspan="4">1</td><td colspan="47">10614141123459</td></tr>
<tr><td>**0**</td><td>**0**</td><td>**0**</td><td>**0**</td><td>**0**</td><td>**0**</td><td>**0**</td><td>**1**</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table>

3.6.1.2 Binary encoding example for GTIN-8: 95200002 using 'AA' compression header

<table>
<tr><td colspan="8">GS1 AI (01) GTIN-8</td><td colspan="27">Value of GTIN-8 is fixed-length numeric
and left-padded to a total of 27 bits = ceiling(8*log(10)/log(2))</td></tr>
<tr><td colspan="4">A</td><td colspan="4">A</td><td colspan="27">95200002</td></tr>
<tr><td>**1**</td><td>**0**</td><td>**1**</td><td>**0**</td><td>**1**</td><td>**0**</td><td>**1**</td><td>**0**</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
</table>

3.6.1.3 Binary encoding example for Expiration Date and Time: (7003)1903111228

GS1 AI (7003)																Value of AI (7003) is fixed-length numeric and left-padded to a total of 34 bits = ceiling (10*log(10)/log(2))																											
7				0				0				3				1903111228																											
0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	0	1	1	0	0	0	0	1	1	1	1	0	0

3.6.2 GS1 Application Identifiers whose values are all-numeric and variable-length

The following example shows binary encoding of GS1 Application Identifiers where the value is both all-numeric but of variable length. In this situation, the GS1 Application Identifier (encoded at four bits per digit) is immediately by L bits for the length indicator, whose binary value corresponds to the actual length of the value string as an integer number of digits, D. If D_{max} is the maximum permitted length for the value of the GS1 Application Identifier, then L is given by the following formula:

$$L = \text{ceiling}(\log(D_{max})/\log(2))$$

The L bits for the length indicator are then populated with the binary value for D, representing the actual length.

The number of bits N for the actual value is given by the formula:

$$N = \text{ceiling}(D * \log(10)/\log(2))$$

The binary value is left-padded (highest significant bits set to '0') in order to reach the total of N bits.

3.6.2.1 Binary encoding example for Count: (30)5000

GS1 AI(30)								Actual Length			Value of AI (30) is variable-length numeric (up to 8 digits) and left-padded to a total of 14 bits = ceiling (4*log(10)/log(2))													
3				0				4			5000												
0	0	1	1	0	0	0	0	1	0	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0

3.6.3 GS1 Application Identifiers whose values are alphanumeric and variable-length

The following examples show binary encoding of GS1 Application Identifiers where the value is permitted to be alphanumeric and of variable length. In this situation, the GS1 Application Identifier (encoded at four bits per digit) is immediately followed by a 3-bit Encoding Indicator whose value indicates whether the actual value can be considered as all-numeric, lower-case hexadecimal, upper-case hexadecimal, characters from a URI-safe base 64 alphabet or ASCII characters. The value of the 3-bit Encoding Indicator is populated according to the table in section *3.4.4.1*. The 3-bit Encoding Indicator is then followed by a Length Indicator consisting of N bits that indicate the length of the actual value in digits or characters.

3.6.3.1 Batch/Lot Number: (10)XYZ*8765

In this example, the value contains a symbol character (asterisk, *) so encoding value 4 is selected to support ASCII characters, using 7 bits for each character of the value.

GS1 AI (10)		Encoding	Actual Length	Value of AI (10) is variable-length alphanumeric (up to 20 characters) encoded for this example value using 7 bits per ASCII character, i.e. 8 characters * 7 bits per character = 56 bits							
1	0	4	8	X	Y	Z	*	8	7	6	5
0 0 0 1	0 0 0 0	1 0 0	0 1 0 0 0	1 0 1 1 0 0 0	1 0 1 1 0 0 1	1 0 1 1 0 1 0	0 1 0 1 0 1 0	0 1 1 1 0 0 0	0 1 1 0 1 1 1	0 1 1 0 1 1 0	0 1 1 0 1 0 1

3.6.3.2 Batch/Lot Number: (10)XYZ98765

In this example, the value contains only alphanumeric characters, so encoding value 3 is selected to support characters from the URI-safe base 64 alphabet, using 6 bits for each character of the value.

GS1 AI (10)		Encoding	Actual Length	Value of AI (10) is variable-length alphanumeric (up to 20 characters) encoded for this example value using 6 bits per URI-safe base64 character, i.e. 8 characters * 6 bits per character = 48 bits							
1	0	3	8	X	Y	Z	9	8	7	6	5
0 0 0 1	0 0 0 0	0 1 1	0 1 0 0 0	0 1 0 1 1 1	0 1 1 0 0 0	0 1 1 0 0 1	1 1 1 1 0 1	1 1 1 1 0 0	1 1 1 0 1 1	1 1 1 0 1 0	1 1 1 0 0 1

3.6.3.3 Batch/Lot Number: (10)ABC98765

In this example, the value contains only numeric digits and upper-case characters A-F, so encoding value 2 is selected to support upper case hexadecimal characters, using 4 bits for each character of the value.

GS1 AI (10)		Encoding	Actual Length	Value of AI (10) is variable-length alphanumeric (up to 20 characters) encoded for this example value using 4 bits per upper case hexadecimal character, i.e. 8 characters * 4 bits per character = 32 bits							
1	0	2	8	A	B	C	9	8	7	6	5
0 0 0 1	0 0 0 0	0 1 0	0 1 0 0 0	1 0 1 0	1 0 1 1	1 1 0 0	1 0 0 1	1 0 0 0	0 1 1 1	0 1 1 0	0 1 0 1

3.6.3.4 Batch/Lot Number: (10)12398765

In this example, the value contains only numeric digits, so encoding value 0 is selected to support integer encoding, using N bits for the value, where N is related to the actual length L by the formula:

$$N = \text{ceiling}(L * \log(10)/\log(2))$$

The binary value is left-padded to reach a total of N bits.

GS1 AI (10)		Encoding	Actual Length	Value of AI (10) is variable-length alphanumeric (up to 20 characters) encoded for this example value as an integer, with left-padding to total of 27 bits
1	0	0	8	12398765
0 0 0 1	**0 0 0 0**	0 0 0	0 1 0 0 0	0 0 0 1 0 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 1

3.6.4 Optimisation using pre-defined sequences of GS1 Application Identifiers

3.6.4.1 GTIN + expiry date & time: (01)10614141123459(7003)1903111228

From tableOpt, it is possible to find a code (‘0E’) that expresses GTIN (01) followed by expiration date and time (7003)

Code: 0000 1110

Value: 00010011010011101001100000111000111101110000011

Value: 0001110001011011110010110000111 1000

Optimisation Code '0E' = (01)...,(7003)...		Value of AI (01) is fixed-length numeric and left-padded to a total of 47 bits = ceiling (14*log(10)/log(2))	Value of AI (7003) is fixed-length numeric and left-padded to a total of 34 bits = ceiling (10*log(10)/log(2))
0	E	10614141123459	1903111228
0 0 0 0	1 1 1 0	0 0 0 1 0 0 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 0 0 0 0 0 1 1	0 0 0 1 1 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 1 0

In this example, 32 bits are saved by not encoding Application Identifier 7003 as 0111 0000 0000 0011 between the second and third Value rows, since the optimisation code value ‘0E’ in tableOpt already indicates that the first value corresponds to (01) and the second value corresponds to (7003). This optimisation results in a saving of 3 characters in the corresponding GS1 Digital Link URI (http://example.org/DhNOmDj3Bji3lh4 vs http://example.org/ARNOmDj3BuAGOLeWHg).

3.7 Formal ABNF grammar for compressed GS1 Digital Link URIs

The GS1 Digital Link URI Syntax standard [DL-URI] defines the formal ABNF grammar for uncompressed GS1 Digital Link URIs. This section extends that definition to provide formal ABNF grammar for partially compressed GS1 Digital Link URIs

The following rule defines the URI-safe base 64 alphabet appearing in section 3.4.1.

```
uriSafeBase64char = DIGIT / ALPHA / "_" / "-"


compressedSegment = 1*uriSafeBase64char
```

Section 3.7.1 defines the formal ABNF grammar for partially compressed GS1 Digital Link URIs. Section 3.7.2 defines the formal ABNF grammar for fully compressed GS1 Digital Link URIs.

Decompression software should test for a partially compressed GS1 Digital Link URI before it tests for a fully compressed GS1 Digital Link URI, since all partially compressed GS1 Digital Link URIs will also match the ABNF grammar for fully compressed GS1 Digital Link URIs but the reverse is not true.

Note also that a partially compressed GS1 Digital Link URI cannot be mistaken for an uncompressed GS1 Digital Link URI for the following reason:

For an uncompressed GS1 Digital Link URI, the final 2N components of the URI path information consist of key:value pairs in which the key is a GS1 application identifier and the first pair within these 2N components has a key that corresponds to a GS1 primary identification key such as GTIN, indicated either using an alphabetic short name such as `gtin` or using the numeric application identifier, e.g. `01` in the case of GTIN.

For a partially compressed GS1 Digital Link URI, the final component of the URI path information is a compressed segment consisting of characters only from the URI-safe base 64 alphabet and immediately preceded by two path components, representing a key:value pair in which the key corresponds to a GS1 primary identification key such as GTIN, indicated either using an alphabetic short name such as `gtin` or using the numeric application identifier, e.g. `01` in the case of GTIN.

After removal of any trailing forward slash from the URI path information, it is possible to count backwards (from right to left) through the URI path components that are separated via forward slash characters. Counting the final component of the URI path information as component number 1 and working from right to left such that the penultimate component is considered as component number 2, then for a partially compressed GS1 Digital Link URI, the component that corresponds to a primary GS1 identification key will always appear as component 3, i.e. two components before the final component of the URI path information, whereas for an uncompressed GS1 Digital Link URI, a component that corresponds to a primary GS1 identification key will always appear as component m where m is an even number. Since 3 is not an even number, it is possible to use this approach to distinguish between a partially compressed GS1 Digital Link URI and an uncompressed GS1 Digital Link URI. This approach is further explained in section 3.9 and flowcharts D1-D4.

3.7.1 Partially compressed GS1 Digital Link URIs

A partially compressed GS1 Digital Link URI includes exactly one primary GS1 identification key and its value in uncompressed format within the URI path information. The final component of the URI path information is a compressed segment consisting of one of more characters from the URI-safe base 64 alphabet. The two penultimate components of the URI path information are the primary GS1 identification key and its value.

An example of a partially compressed GS1 Digital Link URI is shown below:

```
http://example.org/01/05412345000013/EIiDChplbFkzcAMcW5qmg
```

In the example, the final component of the URI path information is the compressed segment ‘`EIiDChplbFkzcAMcW5qmg`’ consisting of characters only from the URI-safe base 64 alphabet.

The previous two components are '01' (a primary GS1 identification key) and its value, '05412345000013'.

The following rule defines a set of primary GS1 identification keys, referring to definitions appearing in the GS1 Digital Link URI Syntax standard [DL-URI].

```
primaryIDcomponent = gtin-comp / itip-comp / gmn-comp / cpid-comp
                   / gln-comp / partyGln-comp
                   / gsrnp-comp / gsrn-comp / gcn-comp / sscc-comp
                   / gdti-comp / ginc-comp / gsin-comp / grai-comp
                   / giai-comp


partiallyCompressedGS1webURIPath = primaryIDcomponent "/" compressedSegment


partiallyCompressedGS1webURIPattern
           = partiallyCompressedGS1webURIPath [queryStringComp]


partiallyCompressedReferenceGS1webURI
           = "https://id.gs1.org"  partiallyCompressedGS1webURIPattern


partiallyCompressedCustomGS1webURI
           = customURIstem  partiallyCompressedGS1webURIPattern
```

3.7.2 Fully compressed GS1 Digital Link URIs

A fully compressed GS1 Digital Link URI includes a compressed segment as the final component of its URI path information. The final component of the URI path information is a compressed segment consisting of one of more characters from the URI-safe base 64 alphabet.

An example of a partially compressed GS1 Digital Link URI is shown below:

```
http://example.org/DgnYUc1gmji3NU0IREGFDTK2LJm
```

In the example, the final component of the URI path information is the compressed segment 'DgnYUc1gmji3NU0IREGFDTK2LJm' consisting of characters only from the URI-safe base 64 alphabet.

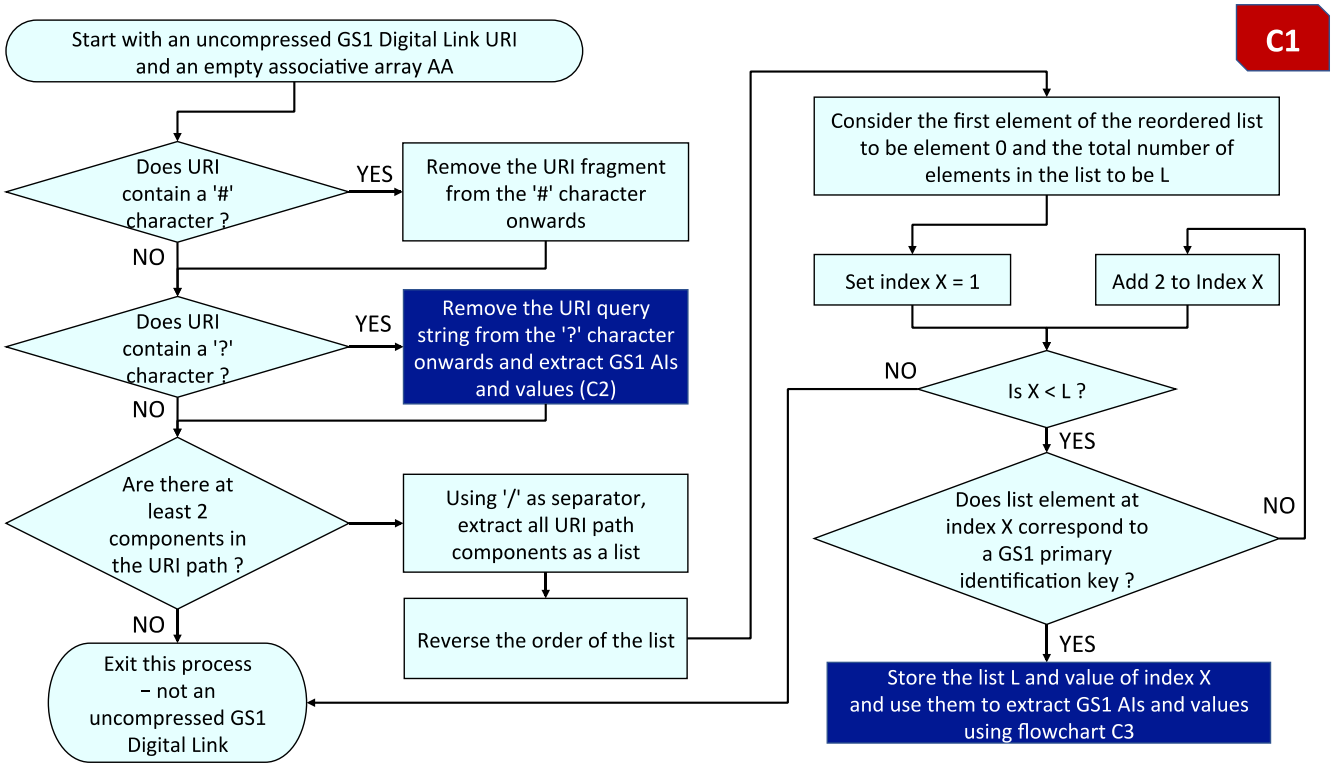
```
fullyCompressedGS1webURIPattern
           = compressedSegment [queryStringComp]


fullyCompressedReferenceGS1webURI
           = "https://id.gs1.org"  fullyCompressedGS1webURIPattern


fullyCompressedCustomGS1webURI
     = customURIstem  fullyCompressedGS1webURIPattern
```


3.8 Compression procedure and flowcharts

Version 1.2 of the GS1 Digital Link standard introduces a new feature, the ability to express an EPC binary string within a compressed GS1 Digital Link URI by making used of two of the previously reserved compression headers. Please refer to section 3.5.1 for further details and examples.

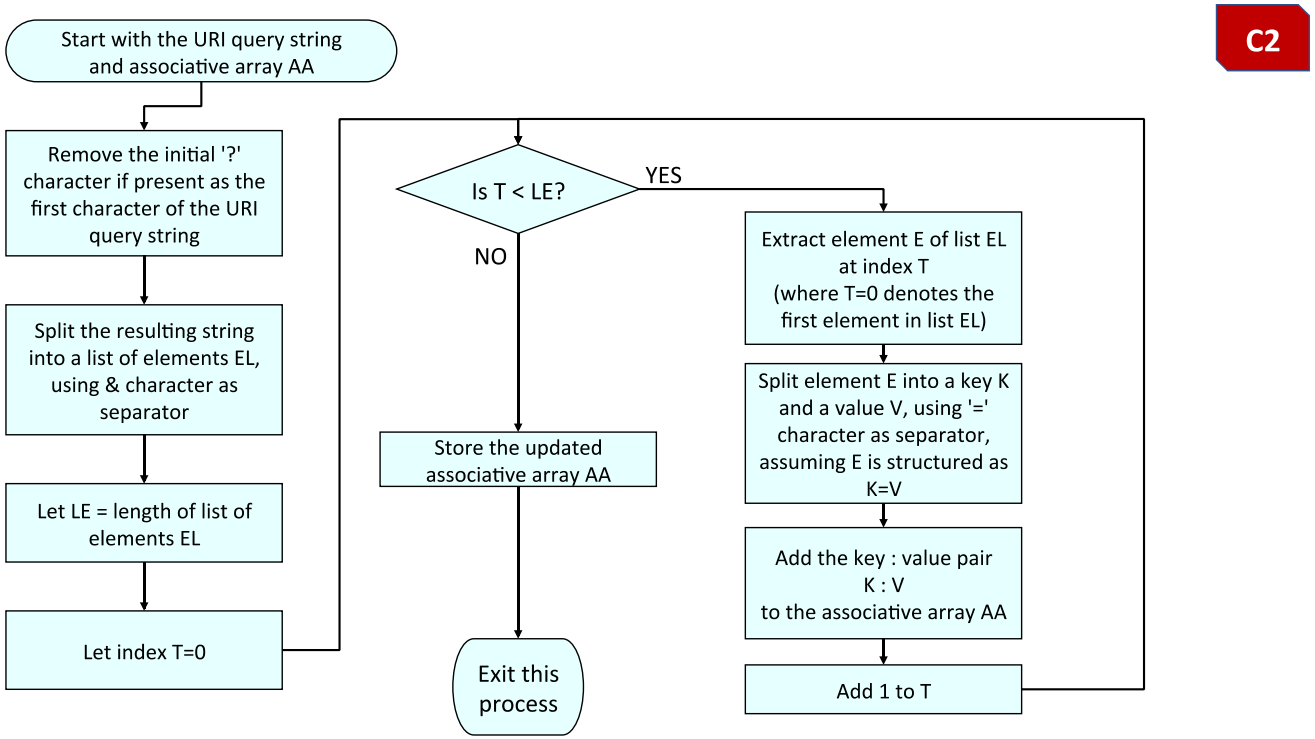
The remainder of this section provides a set of flowcharts to describe the compression procedure to obtain fully or partially compressed GS1 Digital Link URIs when starting from a fully compressed or partially compressed GS1 Digital Link URI.

Figure 3-1 C1: Initial high-level flowchart for parsing uncompressed GS1 Digital Link URIs



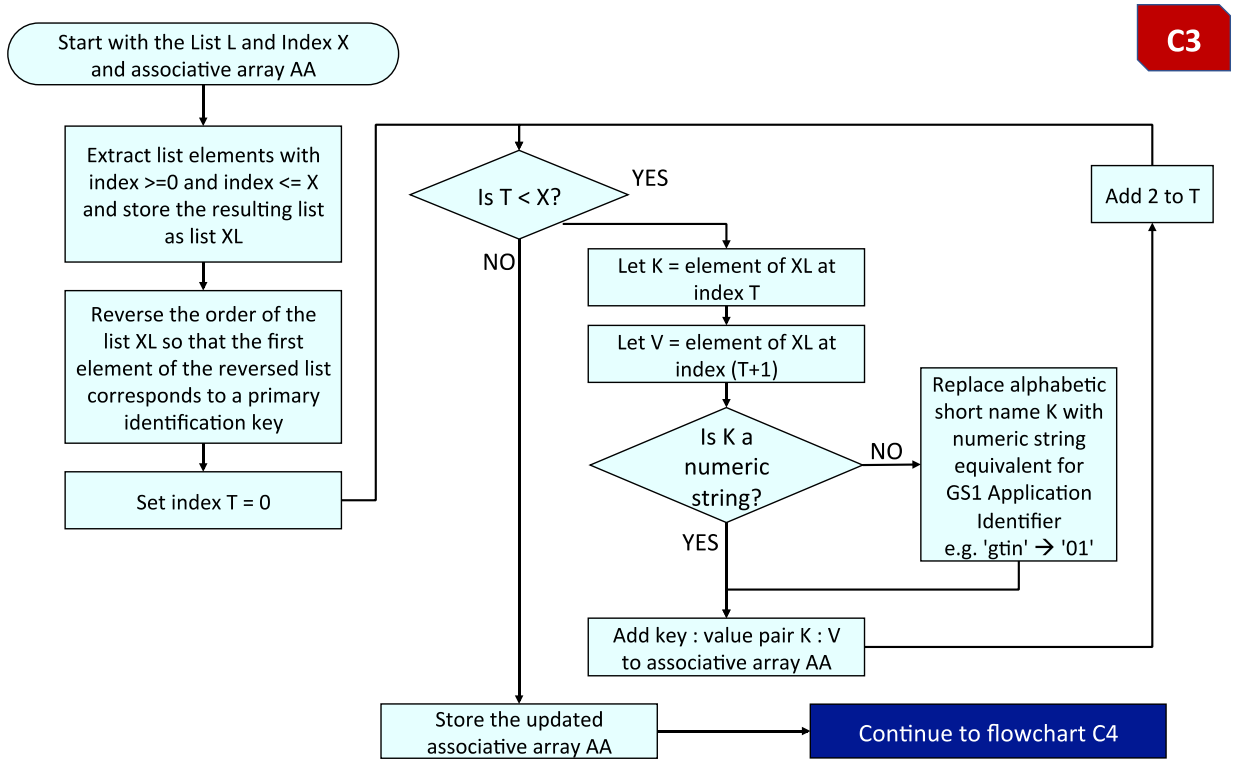
Flowchart C1 explains how to extract GS1 Application Identifiers and their values from an uncompressed GS1 Digital Link URI.

Figure 3-2 C2: Extraction of GS1 Application Identifiers and their values from the URI query string



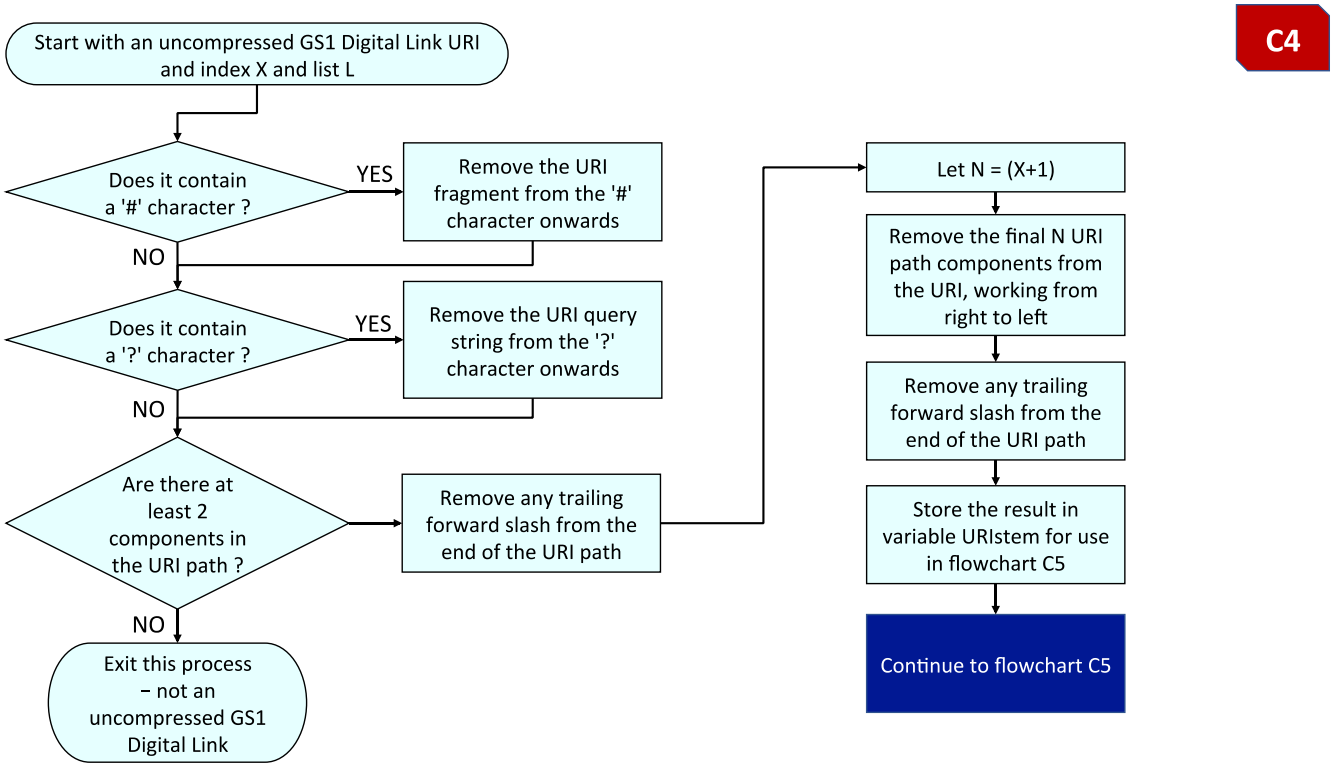
Flowchart C2 explains how to extract GS1 Application Identifiers and their values from the URI query string.

Figure 3-3 C3: Extraction of GS1 Application Identifiers and their values from the URI path information



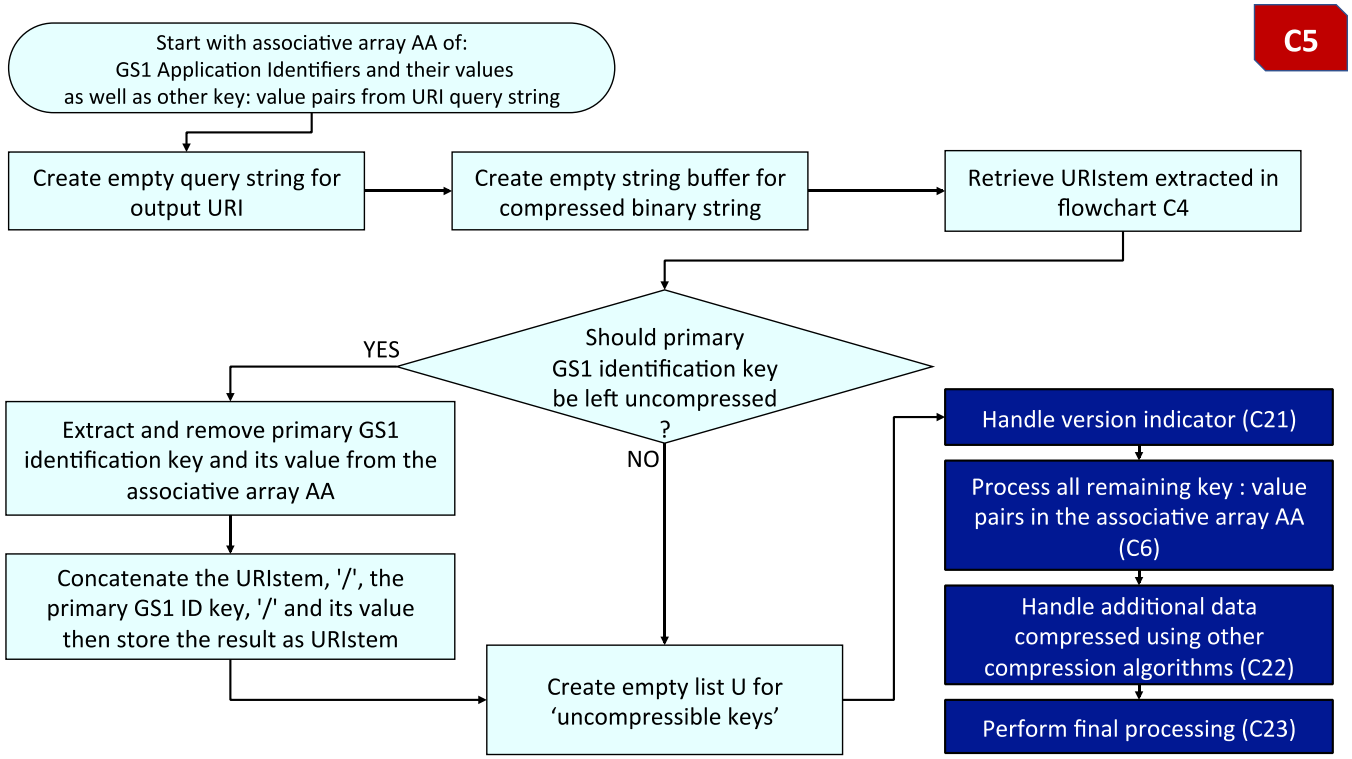
Flowchart C3 explains how to extract GS1 Application Identifiers and their values from the URI path information.

Figure 3-4 C4: Extraction of the URI stem preceding the primary GS1 identification key



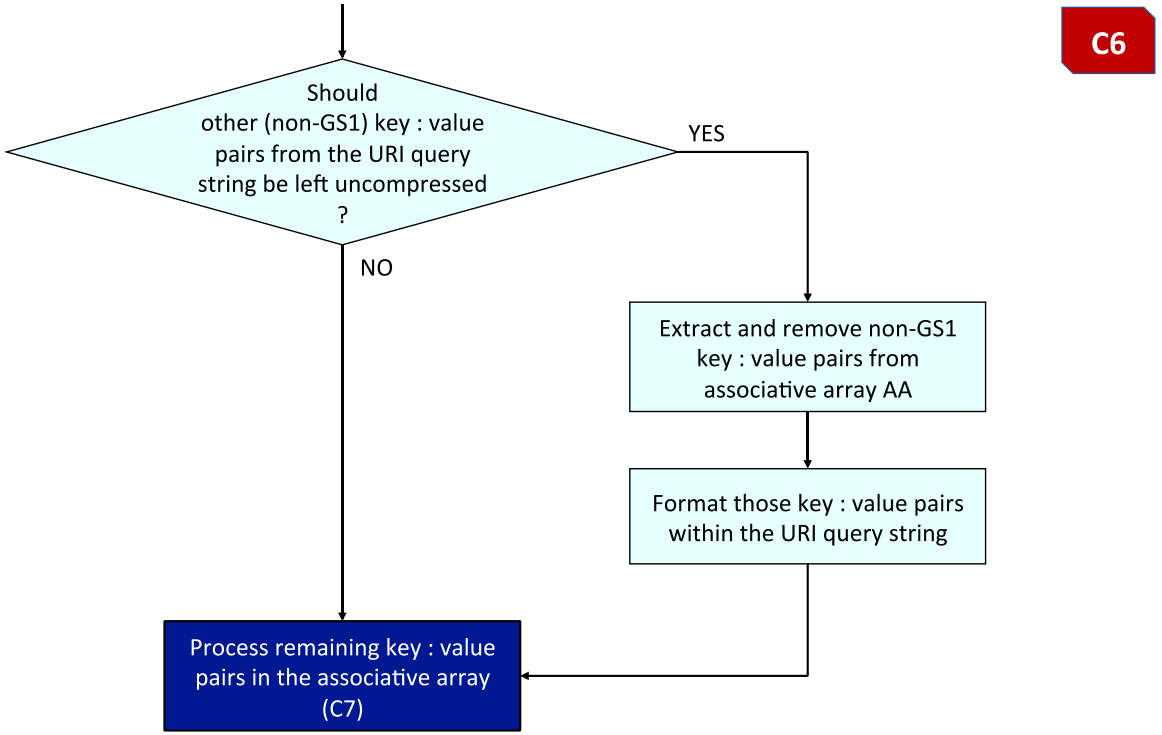
Flowchart C4 explains how to extract the URI stem that precedes the primary GS1 identification key.

Figure 3-5 C5: High-level flowchart for compression of GS1 Digital Link URIs



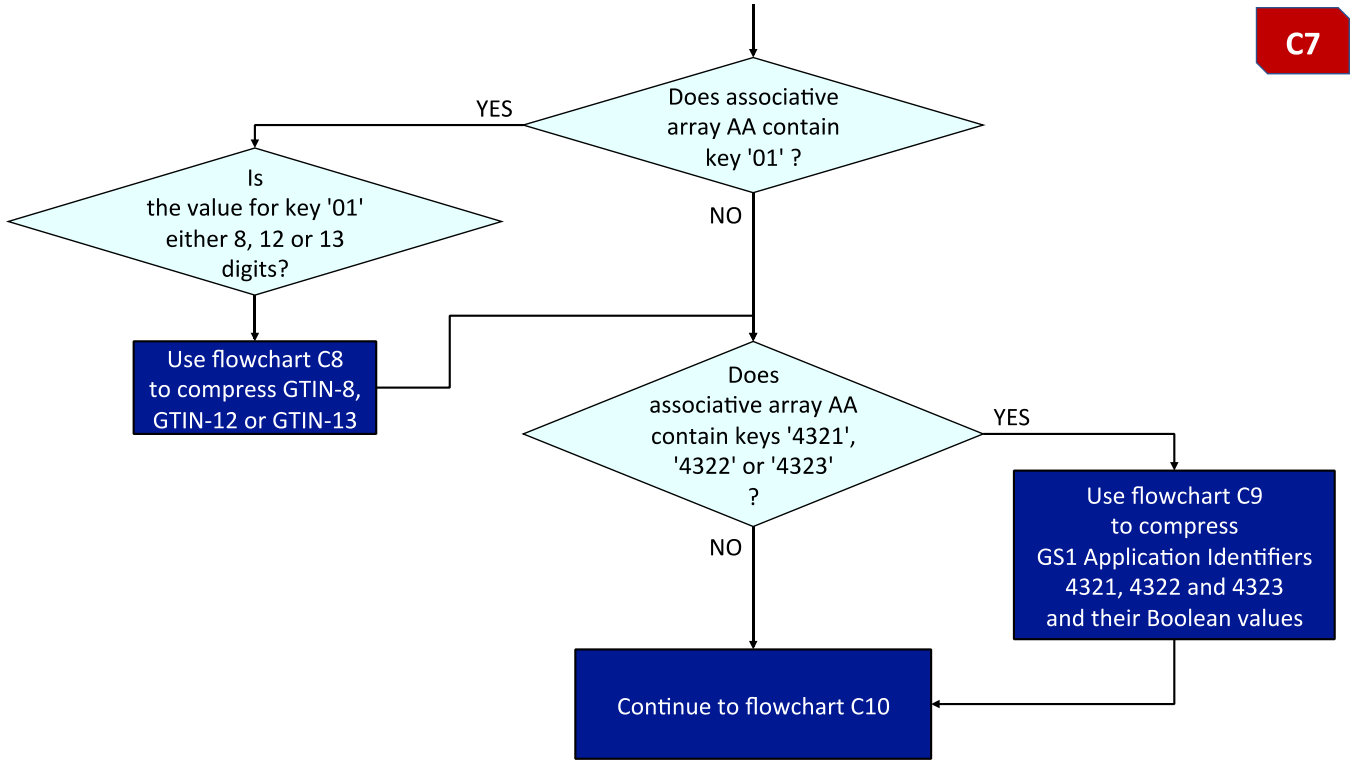
Flowchart C5 provides the high-level flowchart for compression of GS1 Digital Link URIs, starting with an associative array of key:value pairs from element strings or other URI query string key:value pairs. The first decision is whether the primary identifier and its value should be left uncompressed (as is the case for a partially compressed GS1 Digital Link URI). Flowcharts C21, C6, C22 and C23 and their dependents are referenced for further details.

Figure 3-6 C6 : Support non-GS1 key:value pairs from URI query string being compressed or left uncompressed in the URI query string



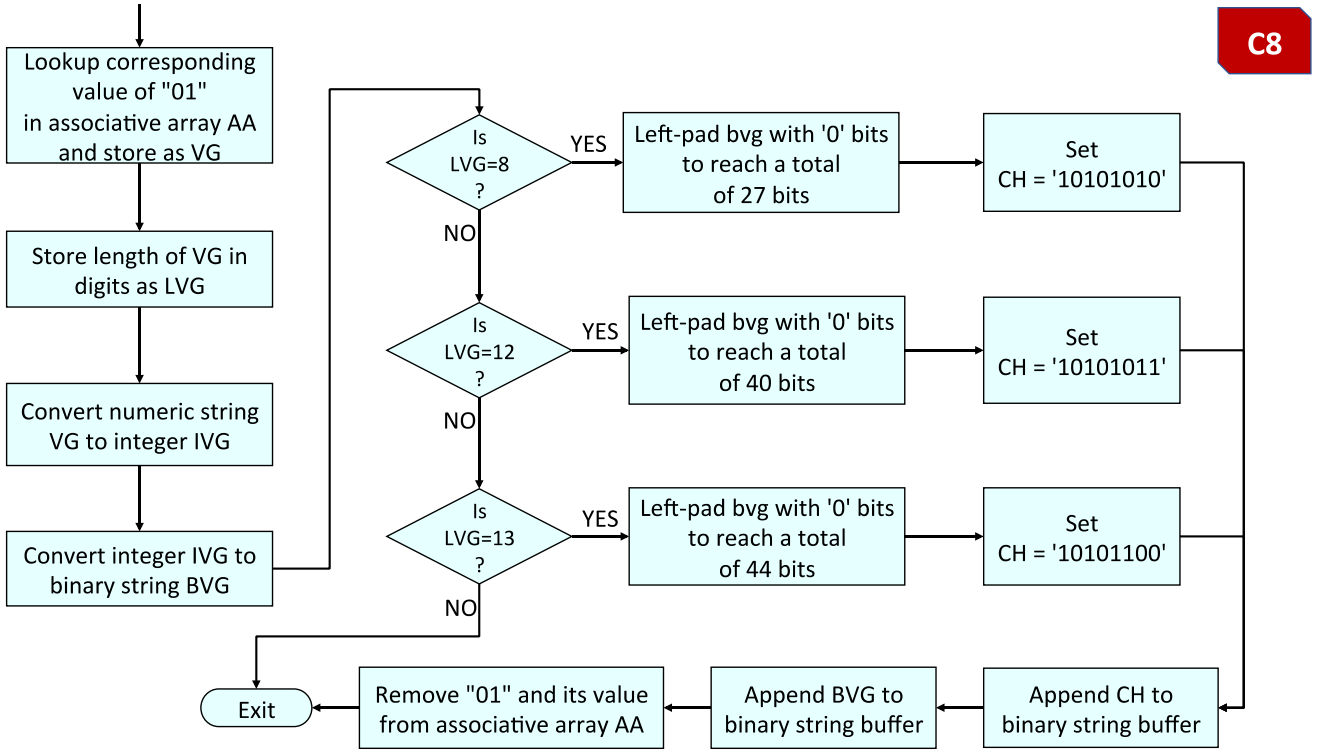
Flowchart C6 explains that if non-GS1 key:value pairs from the URI query string should be compressed, then these must be extracted from the URI query string and included within the associative array. Further processing then moves to flowchart C7.

Figure 3-7 C7: High-level flowchart for special handling of GTIN-8, GTIN-12, GTIN-13 or GS1 Application Identifiers (4321), (4322) or (4323)



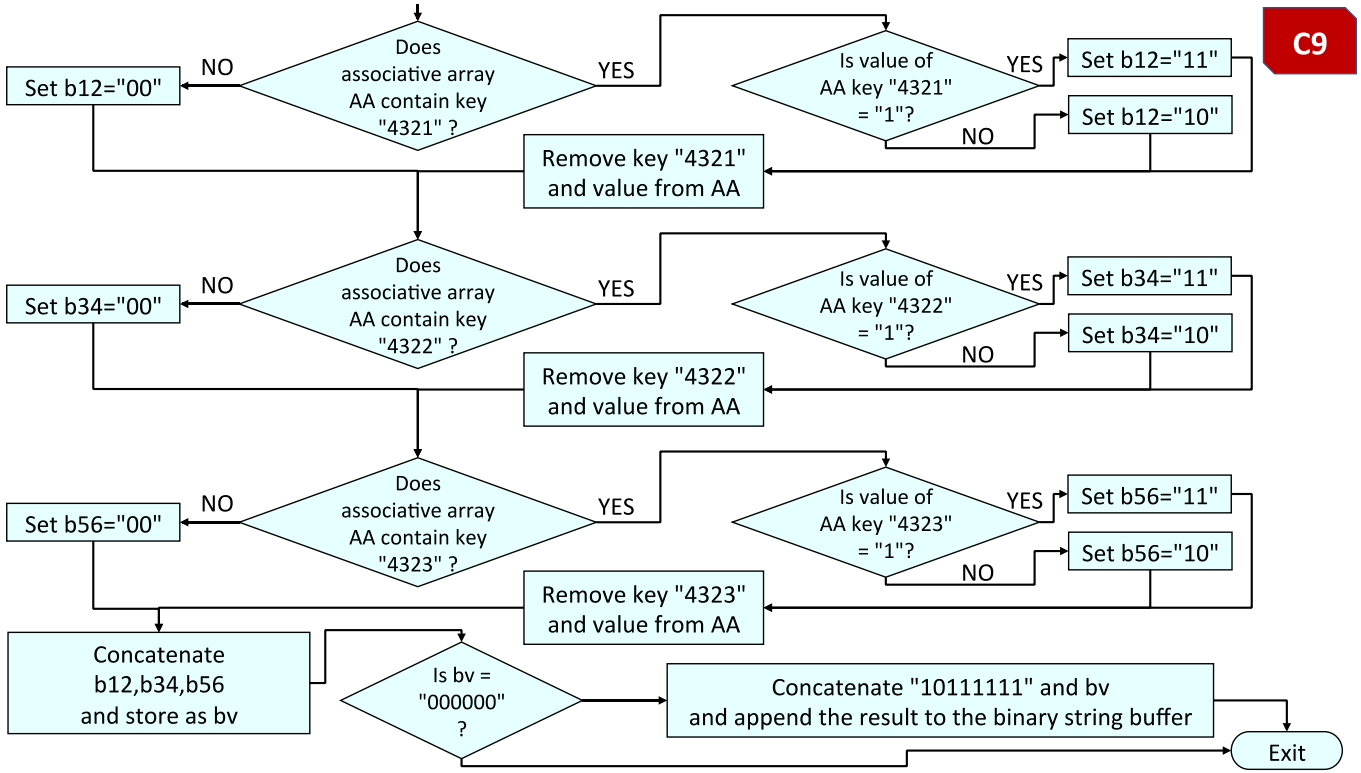
In Flowchart C7, if the associative array AA contains the key '01' and if the length of the GTIN value is 8, 12 or 13 digits, Flowchart C8 is used to compress a GTIN-8, GTIN-12 or GTIN-13 using compression headers "AA", "AB", "AC". If the associative array AA contains any of keys '4321', '4322' or '4323', Flowchart C9 is used to compress the Boolean values of these three GS1 Application Identifiers into a single 6-bit field. Processing then continues at Flowchart C10.

Figure 3-8 C8: Handle compression of GTIN-8, GTIN-12 or GTIN-13 while preserving their length



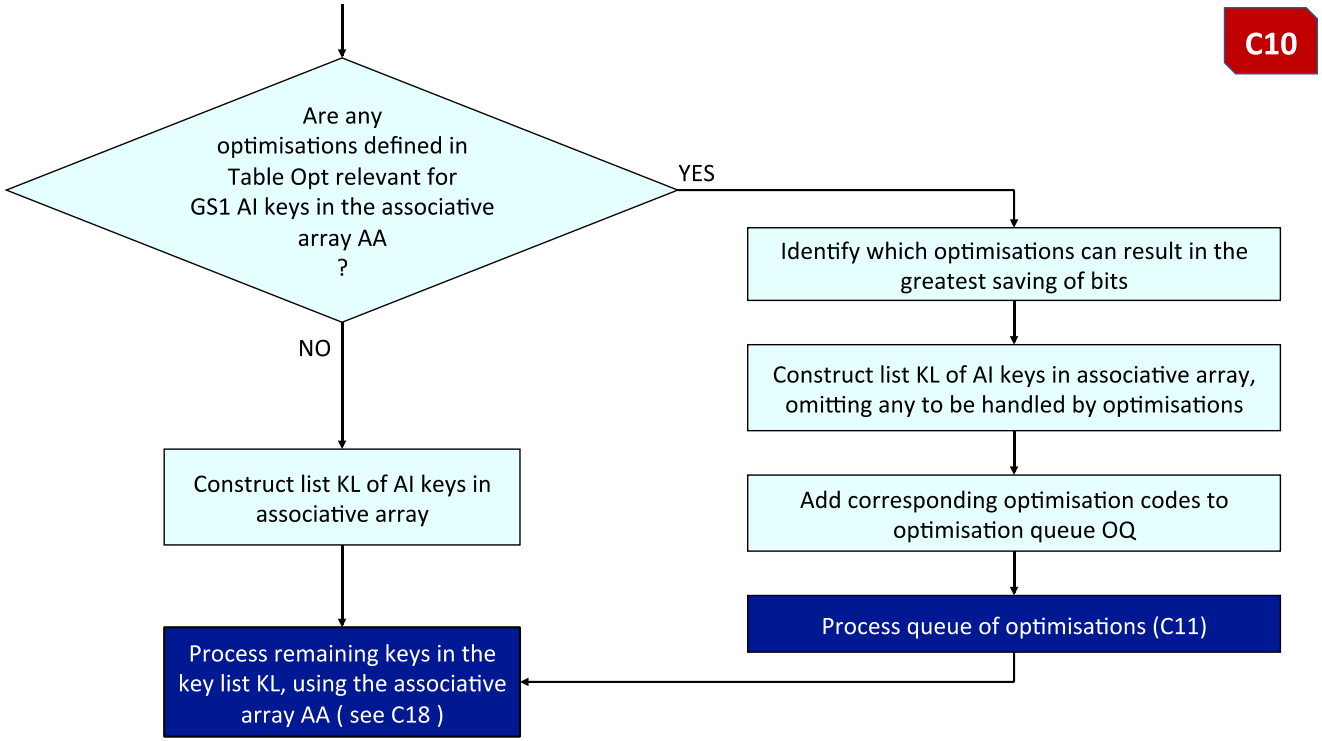
Flowchart C8 explains how to compress a GTIN-8, GTIN-12 or GTIN-13 using the compression headers "AA", "AB","AC" respectively, as explained in section 3.5.3.

Figure 3-9 C9: Handle compression of Boolean values for GS1 Application Identifiers (4321), (4322), (4323)



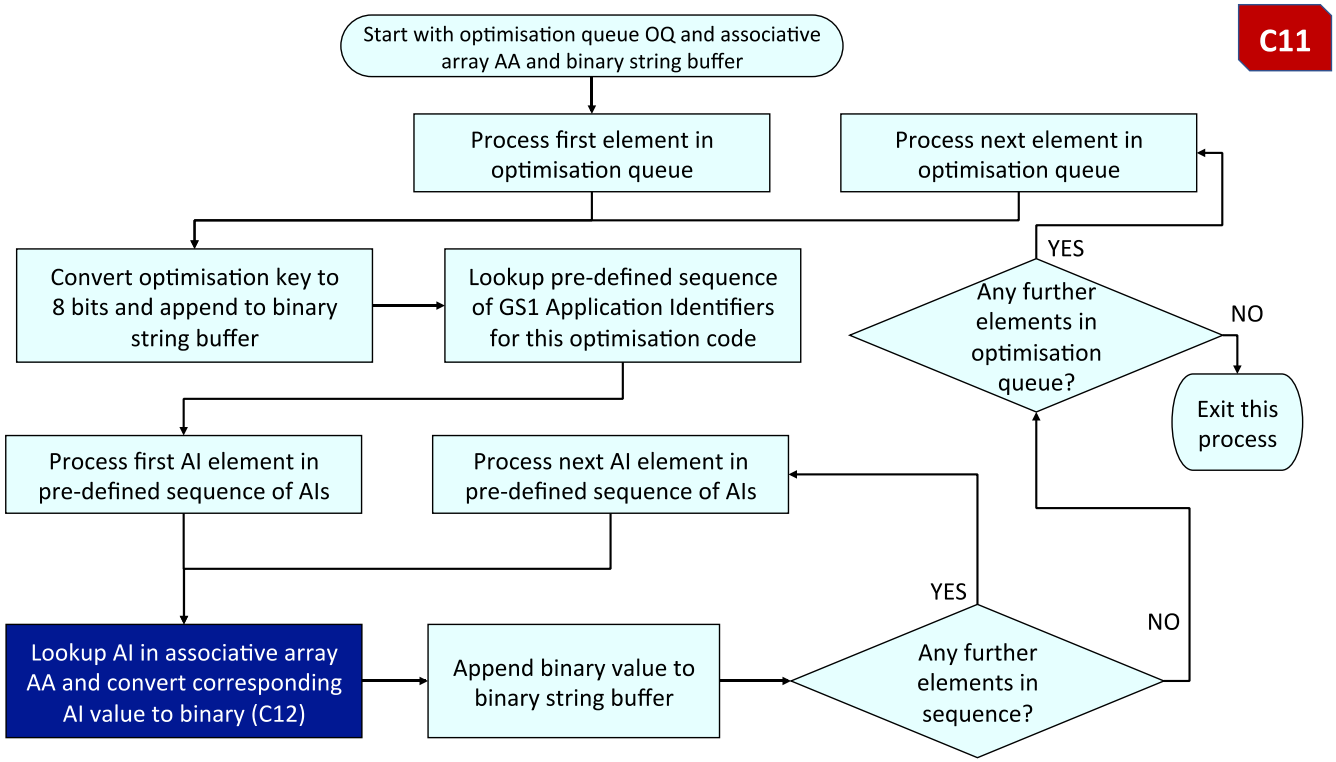
Flowchart C9 handles compression of three GS1 Application Identifiers, (4321), (4322), (4323), each of which has a Boolean value expressed as a single digit (0=false, 1=true). The values of these three AIs are encoded in a single 6-bit field, using compression header "BF". This is explained in further detail in section 3.5.2.

Figure 3-10 C10: Support for optimisations using pre-defined sequences of GS1 Application Identifiers – refers to flowchart C11 for further detail



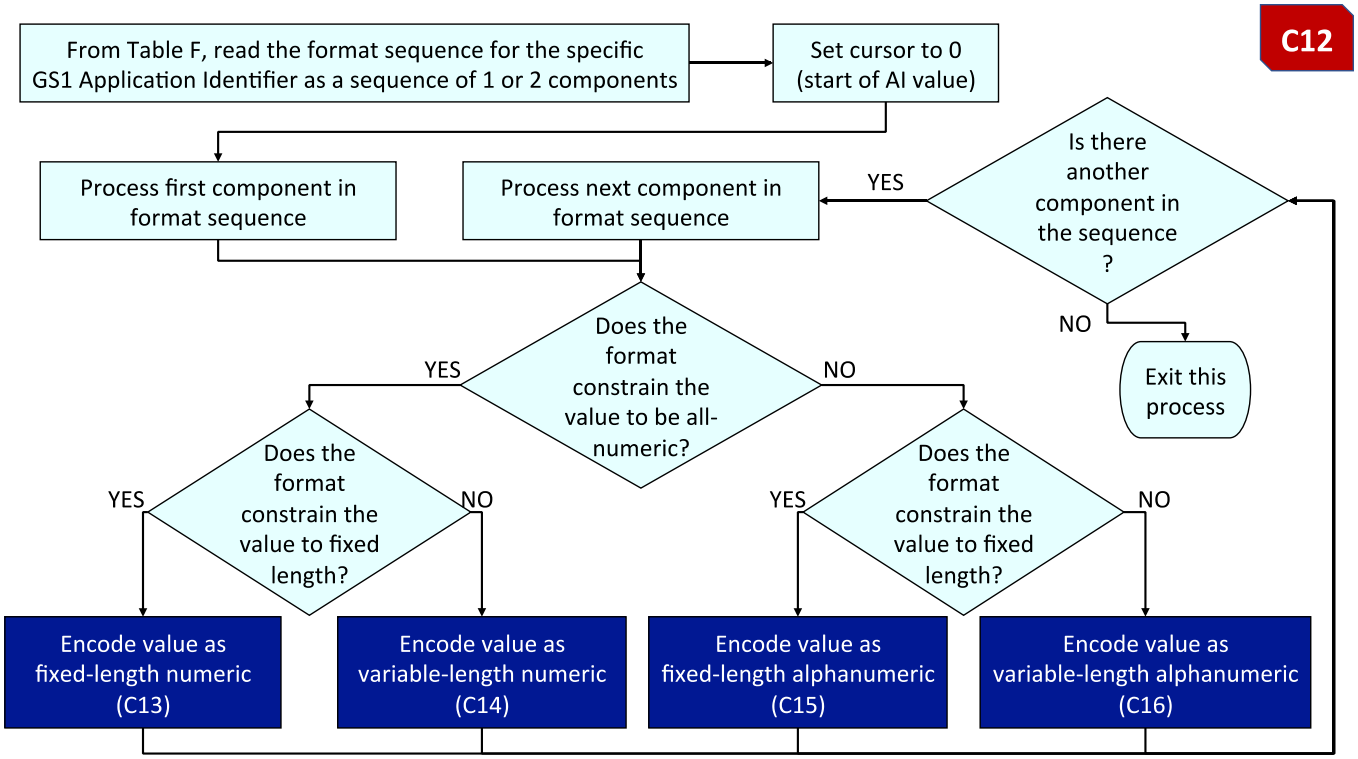
In Flowchart C10, the keys within the associative array are compared against available optimised pre-defined sequences of GS1 AIs, defined in Table Opt. If such optimisations are found, the combination of optimisations that results in the largest total saving of bits is selected and an updated list of AI keys is prepared, omitting those which will be handled via optimisation sequences. Flowchart C11 provides further detail about how to process a queue of such optimisations. Ultimately, any remaining AI keys not handled by optimisations are handled as explained in flowchart C18.

Figure 3-11 C11: Process queue of optimisations for pre-defined sequences of GS1 Application Identifiers



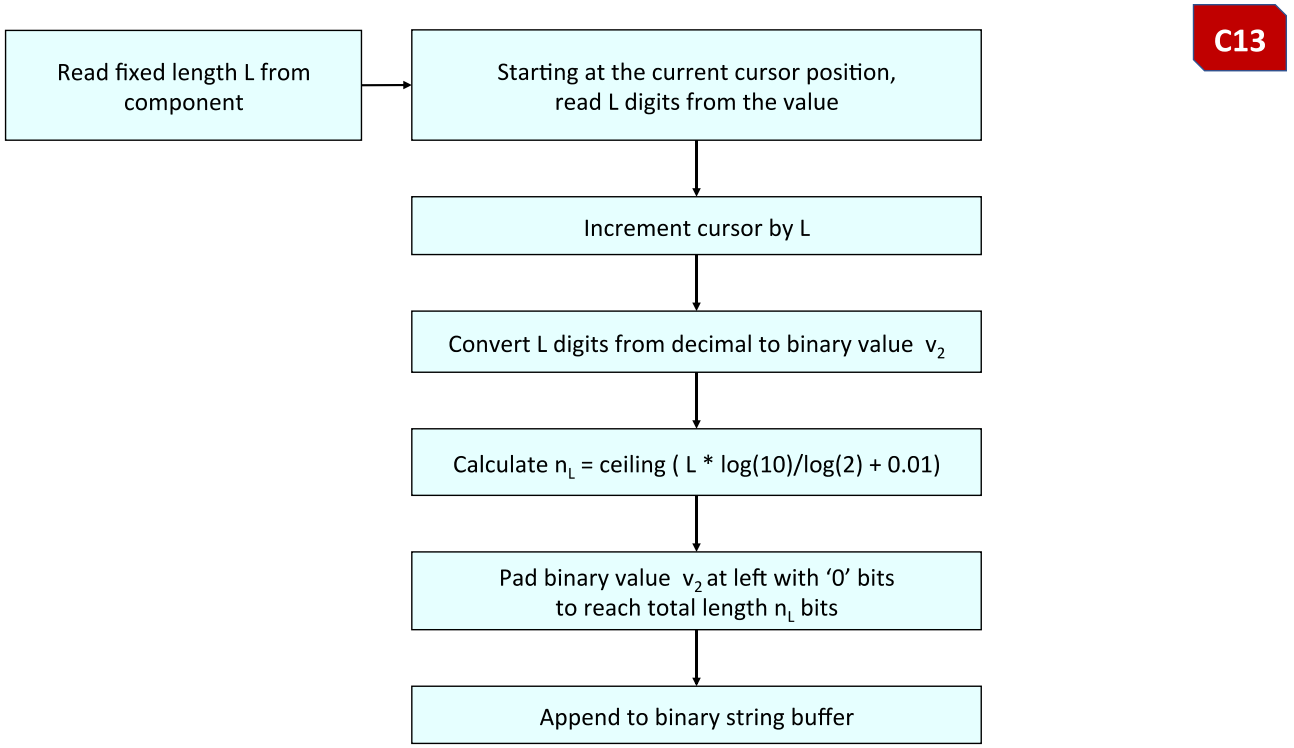
Flowchart C1 explains processing of a queue of optimisations for pre-defined sequences of GS1 Application Identifiers. For each optimisation, the GS1 Application Identifiers are not encoded in binary using 8, 12 or 16 bits per GS1 AI key. Instead a single 8-bit key is used to represent the entire pre-defined sequence. Flowchart C11 contains an outer loop that iterates through all optimisations in the optimisation queue (if more than one exists), while the inner loop iterates through each GS1 AI key defined within one optimised pre-defined sequence. Flowchart C11 references Flowchart C12 for details of how to encode the actual value of each GS1 Application Identifier into the binary string. In this way, a binary string buffer is built up for each optimisation and for each pre-defined AI within each optimisation within the optimisation queue.

Figure 3-12 C12: Convert the value of a GS1 Application Identifier to binary



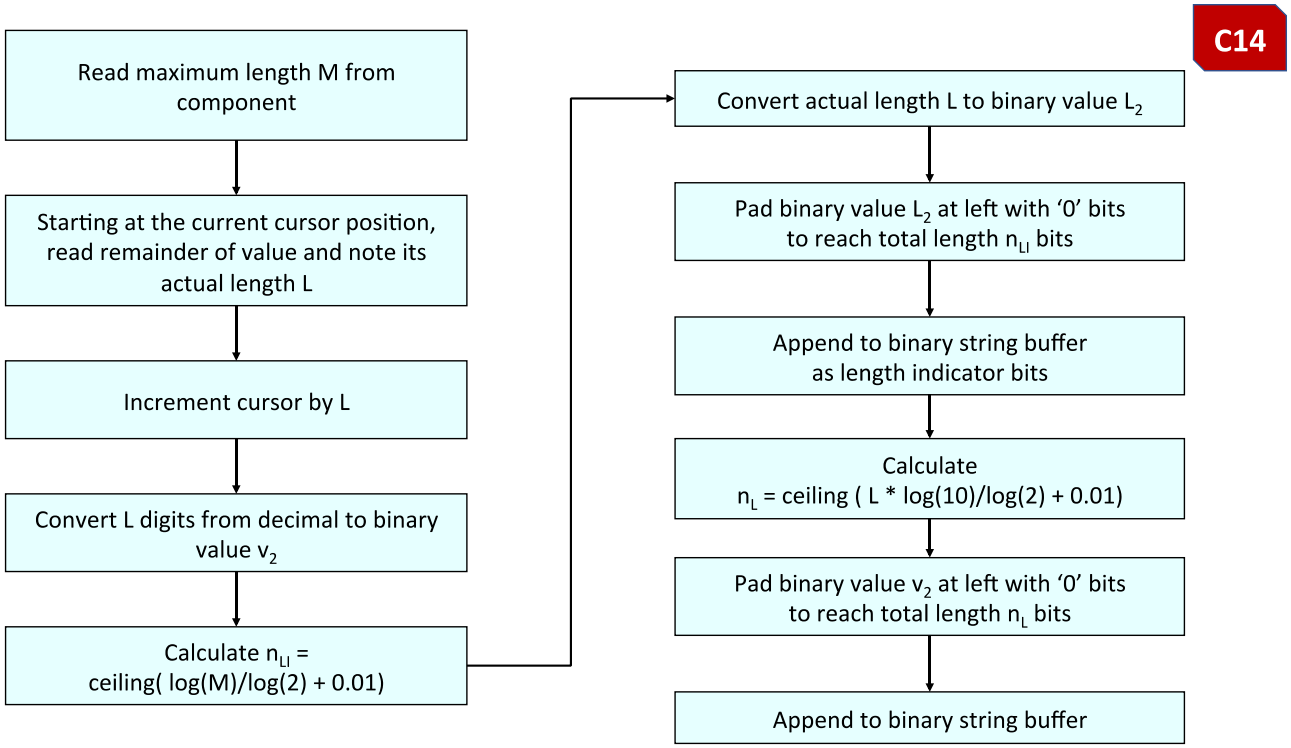
Flowchart C12 explains how the value of each GS1 Application Identifier is encoded in binary. The first step is to find an entry in Table F (the format table) for the GS1 AI key (e.g. 01), which is typically expressed as one or two components, which are processed in turn. For each of these two components, further processing branches depending on whether the format constrained the value to be all-numeric or not and whether the format constrained the value to be fixed length or variable length. Depending on the combination of numeric vs alphanumeric, fixed-length vs variable-length for each component within the GS1 AI value, further processing then refers to additional flowcharts for the encoding of values that are constrained to be fixed-length numeric (C13), variable-length numeric (C14), fixed-length alphanumeric (C15) or variable-length alphanumeric (C16). All current GS1 AI values can be expressed as one or two components and the main loop processes the second component if it is defined in Table F.

Figure 3-13 C13 : Encode a fixed-length numeric value



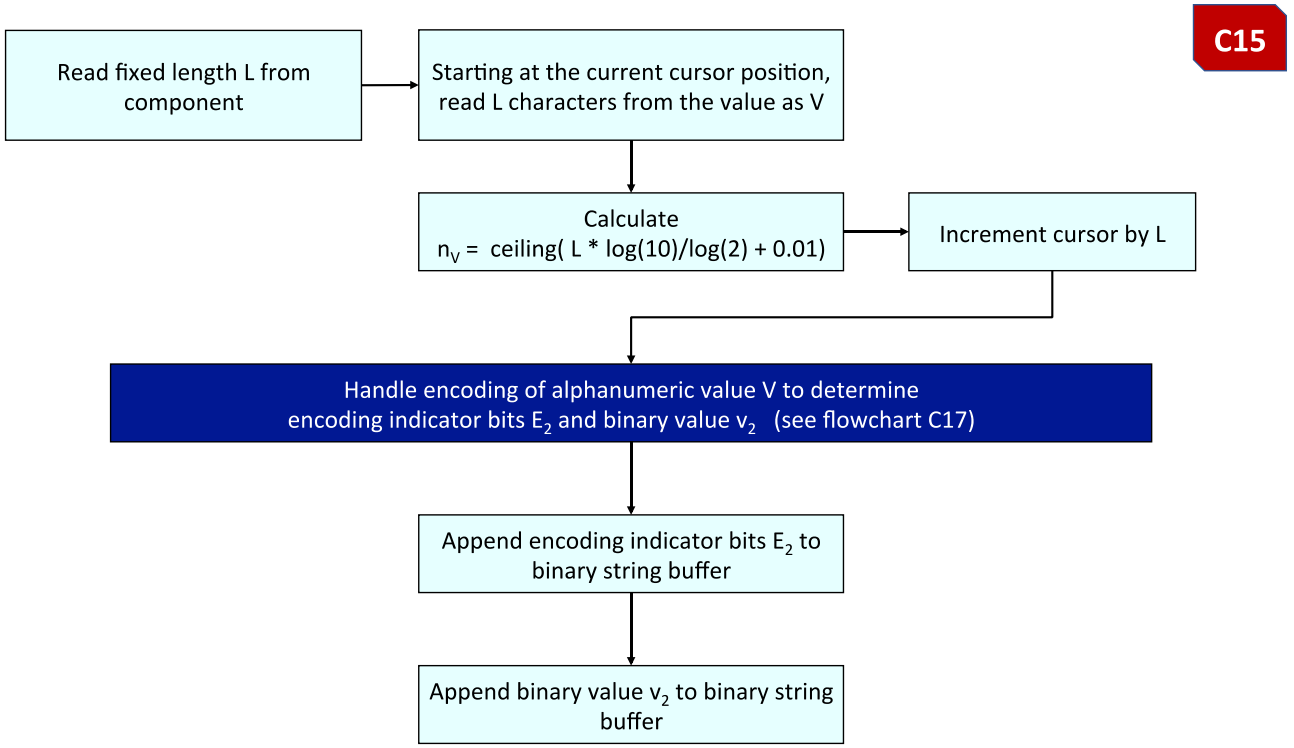
Flowchart C13 explains how to encode a fixed-length numeric value. The number of bits expected is given by the formula for n_L and the binary value is left-padded with '0' bits to reach length n_L.

Figure 3-14 C14: Encode a variable-length numeric value



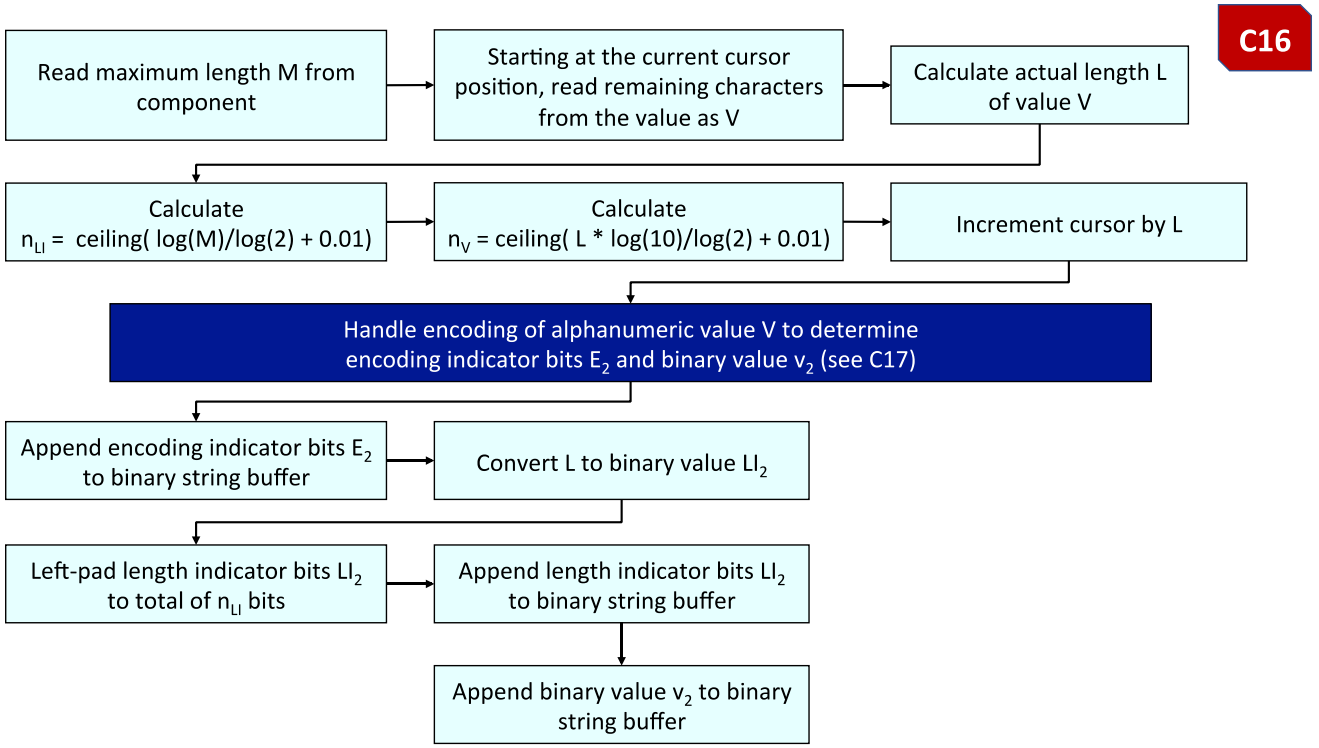
Flowchart C14 explains how to encode a variable-length numeric value. Firstly, a length indicator of an appropriate number of bit n_{LI} is encoded, according to the formula for n_{LI}, encoding the actual length L of the value. Next, the actual value is encoded as a binary integer that is left-padded to The number of bits expected is given by the formula for n_L (based on the maximum allowed length, M) and the binary value is left-padded with '0' bits to reach a total length n_L bits, where n_L is calculated by a different formula based on the actual length of the value.

Figure 3-15 C15: Encode a fixed-length alphanumeric value, referring to flowchart C17 for additional details



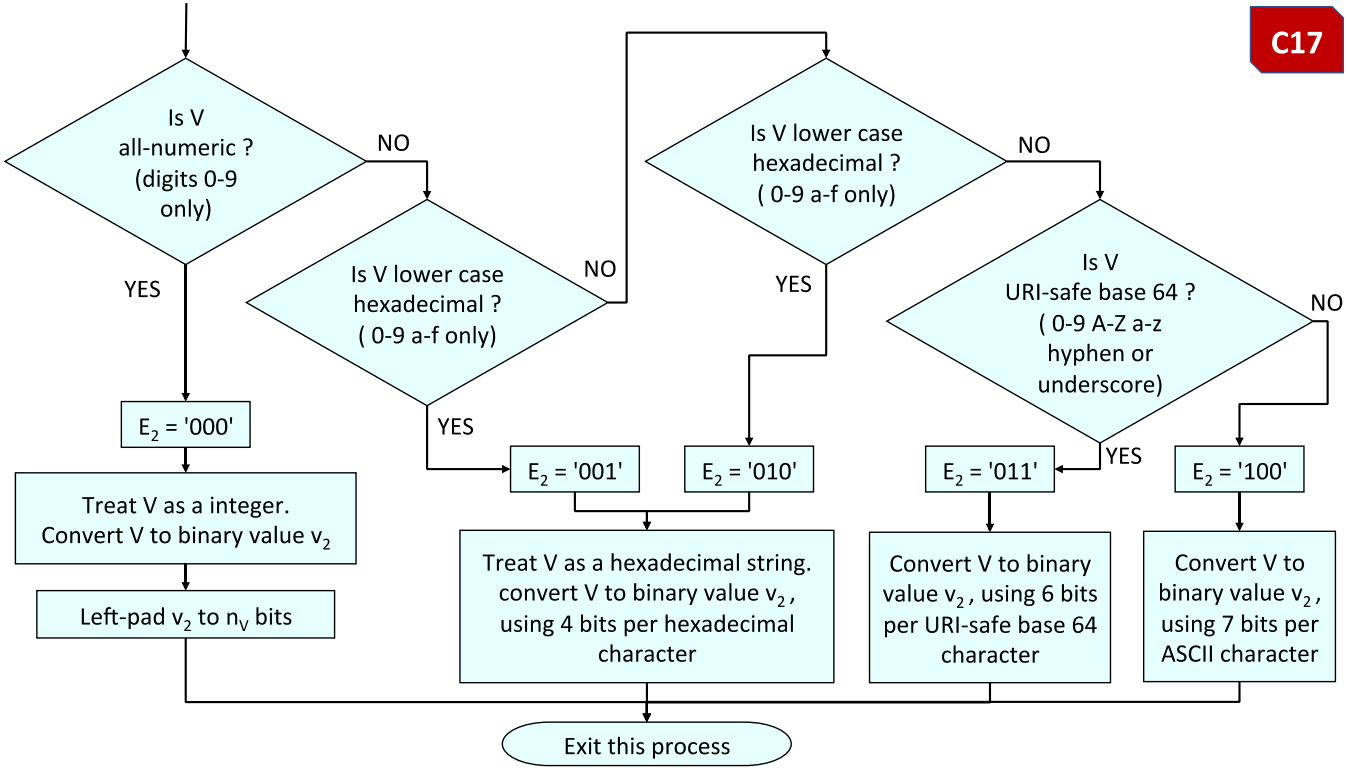
Flowchart C15 explains how to encode a fixed-length alphanumeric value. Flowchart C17 is used to determine an appropriate 3-bit encoding indicator E_2 which is encoded in the binary string before encoding the actual value binary value V_2, using an appropriate number of bits depending on its length and the encoding that was used.

Figure 3-16 C16: Encode a variable-length alphanumeric value, referring to flowchart C17 for additional details



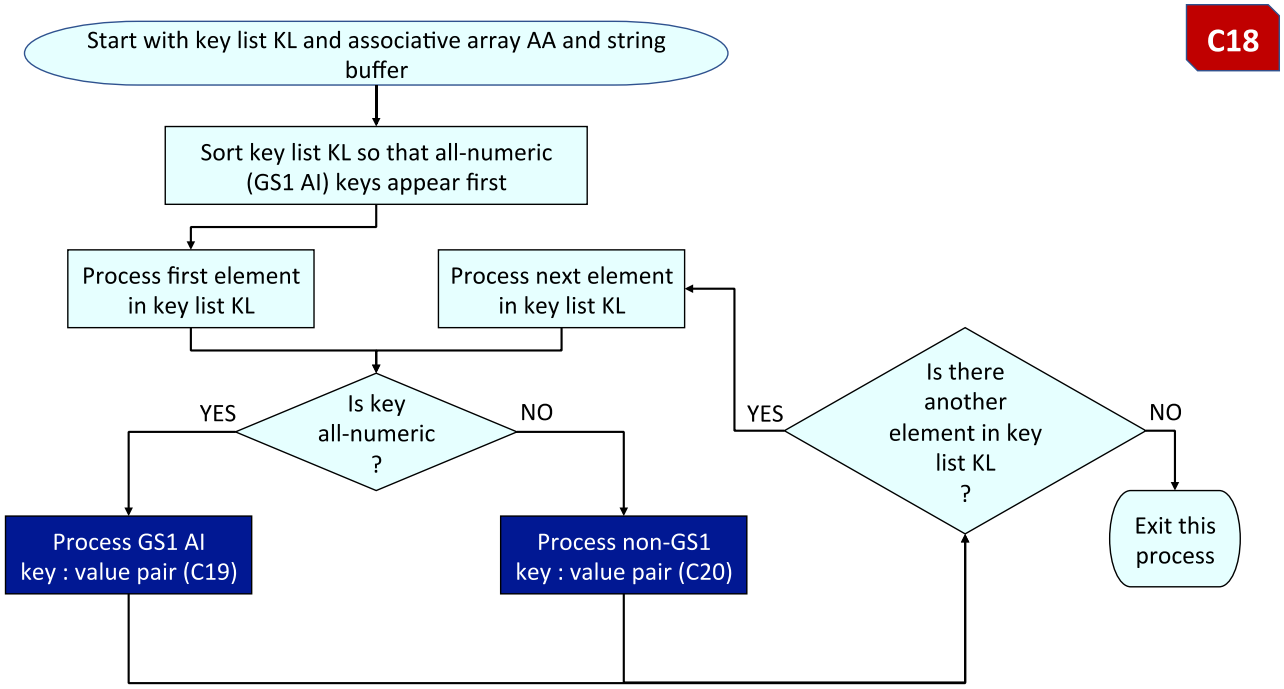
Flowchart C16 explains how to encode a variable-length alphanumeric value. This flowchart combines aspects of flowcharts C14 (variable-length, length indicator) and C15 (encoding indicator). As in Flowchart C15, Flowchart C17 is used to determine the 3-bit encoding indicator E_2 and the appropriate binary encoding of the value V as V_2, depending on the actual value being encoded and its length L. The encoding indicator E_2 is appended to the binary string buffer, followed by the length indicator bits LI_2, followed by the binary value V_2.

Figure 3-2 C17 : Handle binary encoding of alphanumeric values



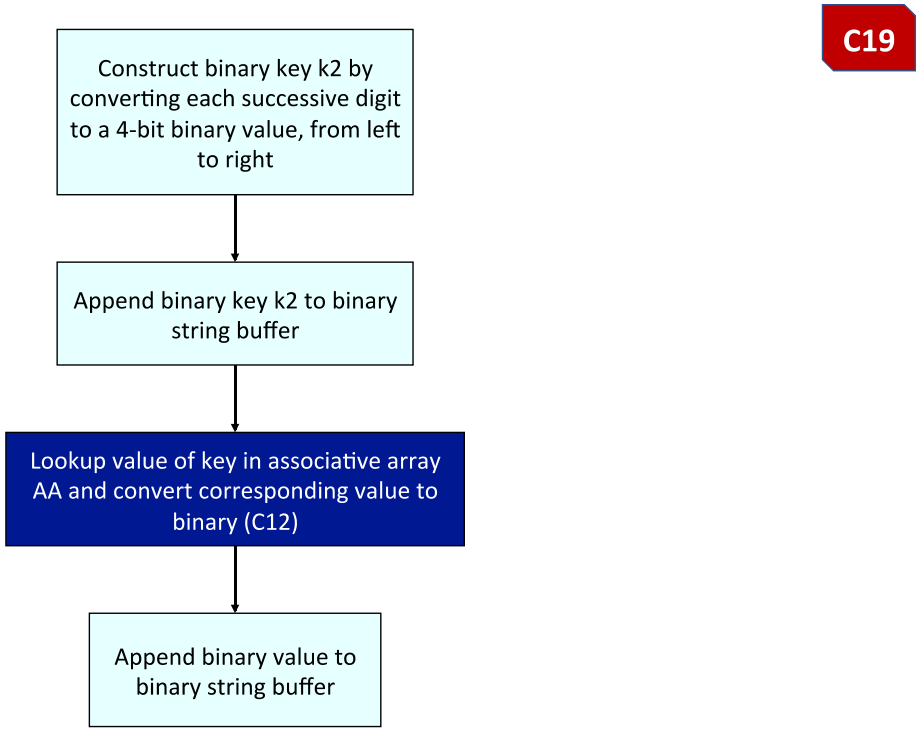
Flowchart C17 explains how to handle binary encoding of an alphanumeric value depending on whether the actual value is all-numeric, lower-case hexadecimal, upper-case hexadecimal, URI-safe base 64 or ASCII. For each of these, an appropriate 3-bit encoding indicator E2 is determined and the actual value V is converted to binary value V_2 and expressed in the appropriate number of bits. For hexadecimal, this requires 4 bits per character. URI-safe base 64 characters use 6 bits per character, while ASCII uses 7 bits per character. For all-numeric values, the value V is converted to binary and left-padded with '0' bits to reach an expected length determined by n_V that was calculated in Flowcharts C15 or C16, since these both reference Flowchart C17.

Figure 3-3 C18: Process remaining keys in the key list KL, using the associative array AA



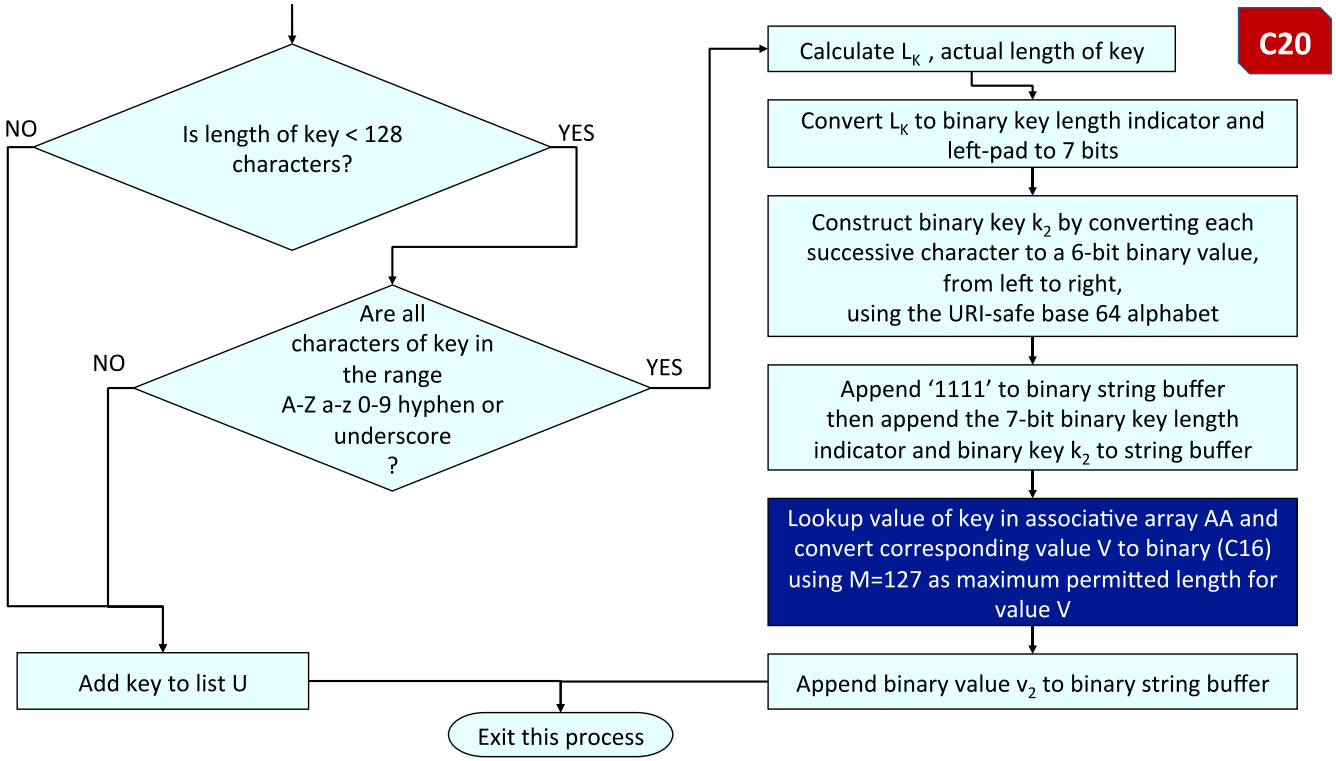
Flowchart C18 explains the processing of each key in the key list KL. The value of each key can be retrieved from the associative array AA. If the key is all-numeric, further processing references Flowchart C19 for processing of a GS1 AI key:value pair. If the key is not all-numeric, further processing references Flowchart C20 for processing of non-GS1 key:value pairs. The main loop continues to process all remaining keys in the key list KL.

Figure 3-4 C19 : Encode GS1 Application Identifier key at 4 bits per digit, then encode the corresponding value in binary, referring to flowchart C12



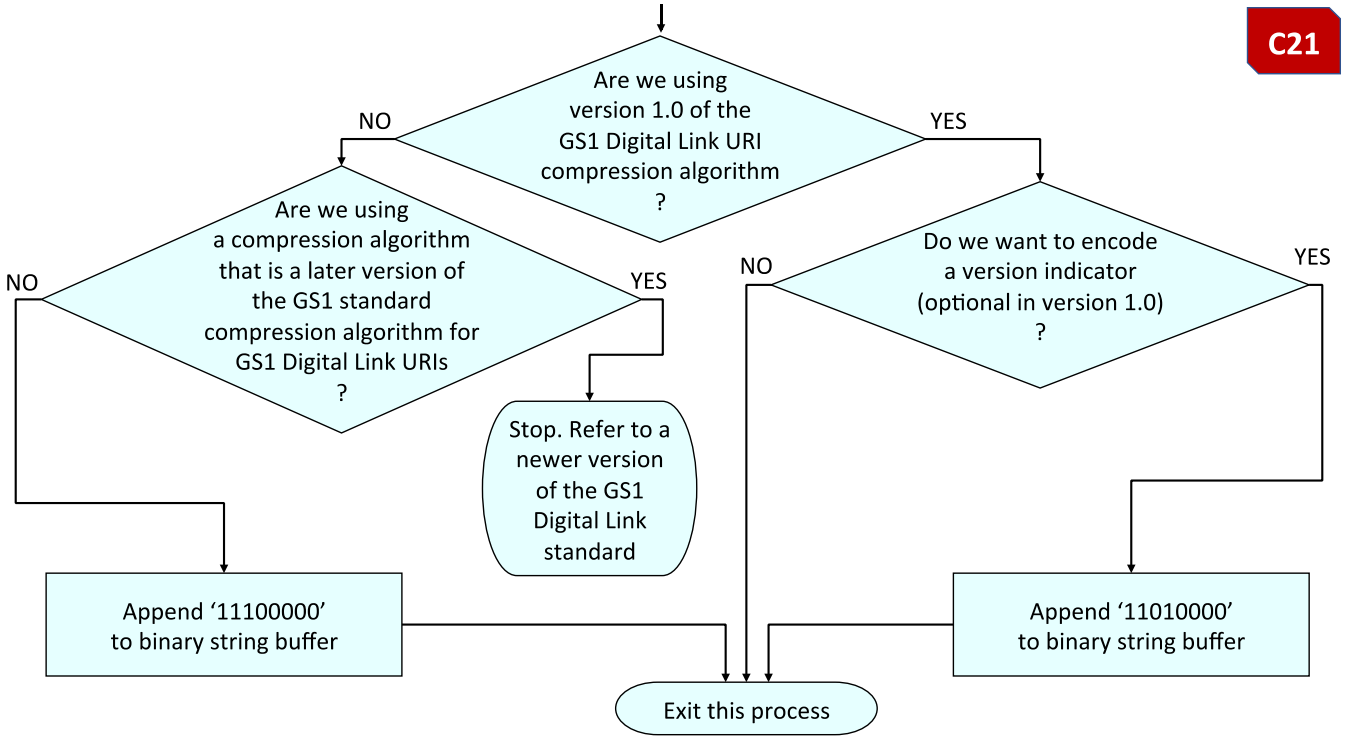
Flowchart C19 explains how to encode each GS1 Application Identifier key as a set of 4 bits per digit. Further processing then references Flowchart C12 and its dependents for formatting of the corresponding value into binary. Finally, the binary value is also appended to the binary string buffer.

Figure 3-20 C20: Support compression of other non-GS1 key:value pairs from URI query string



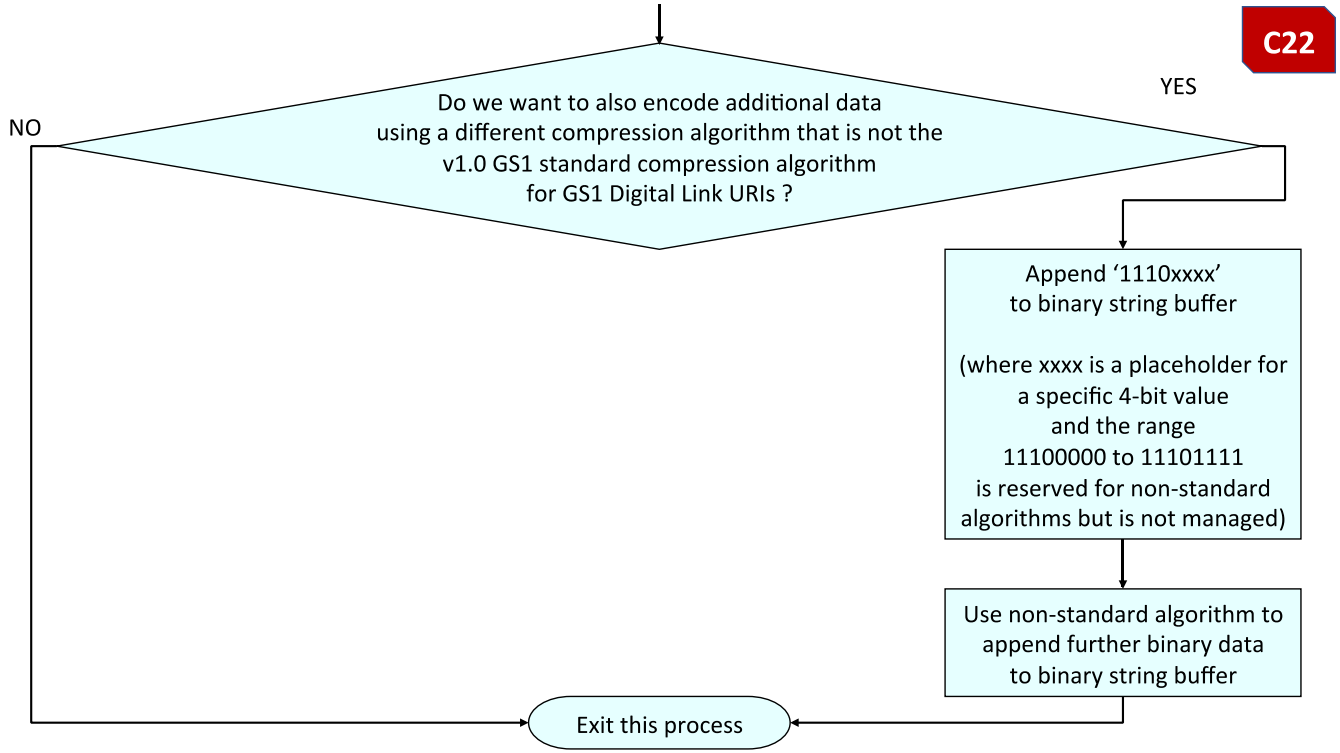
Flowchart C20 explains how to encode non-GS1 key:value pairs within the binary compressed string. For a non-GS1 key to be eligible, it must be less than 128 characters in length and all characters within the key must be within the URI-safe base 64 character set (A-Z a-z 0-9 hyphen and underscore). If either of these conditions are not met, the key is added to list U, to be expressed via the URI query string rather than within the compressed binary string. If both conditions are met, the binary string buffer is encoded with '1111' (as a flag for a non-GS1 key:value pair) followed by a 7-bit length indicator that indicates the actual length of the key, followed by a binary encoding of the key, using the URI-safe base 64 alphabet, at 6 bits per key character. Further processing then references Flowchart C16 for the formatting of the value in binary as v_2 and this is finally appended to the binary string buffer.

Figure 3-21 C21: Handle version indicator (optional in v1.0)



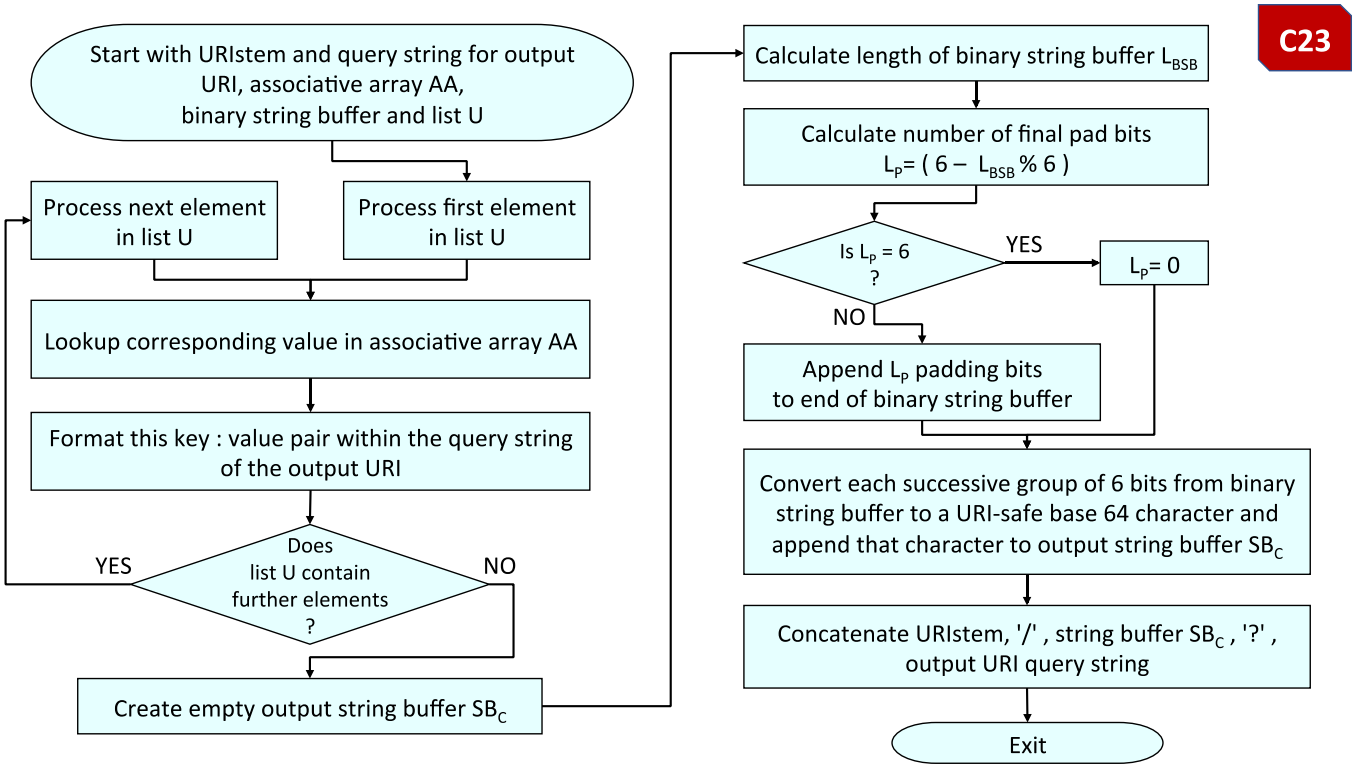
Flowchart C21 explains how to encode a version indicator. For version 1.0 of the GS1 Digital Link URI compression algorithm, the version indicator is optional but if it is encoded, it must be the value 11010000. Note that encoding a value of 11100000 (or more generally 1110xxxx) is a special reserved range that indicates that all bits following the 1110 of 1110xxxx should be considered as belonging to a compression algorithm that is not part of the GS1 Digital Link standard. This represents an extension point or handover point to non-standard compression algorithms and enables a compressed binary string to make use of the GS1 standard compression algorithm first for the encoding of GS1 AIs and non-GS1 key:value pairs from the URI query string, then hand over to a non-standard compression algorithm for storage of other data. If 1110xxxx appears as the first 8 bits of the compressed binary string, the GS1 standard decompression algorithm should stop further processing at this point.

Figure 3-22 C22: Support compression of additional data using a non-standard compression algorithm



Flowchart C22 is related to part of Flowchart C21 and explains that it is possible to use the special reserved sequence 1110xxxx anywhere in the binary string where the initial 8-bit sequence would be read (e.g. to extract a further GS1 AI or optimisation sequence) in order to indicate that everything to the right of 1110 is encoded using a compression algorithm that is not part of the GS1 Digital Link standard.

Figure 3-23 C23: Final assembly of compressed GS1 Digital Link URI

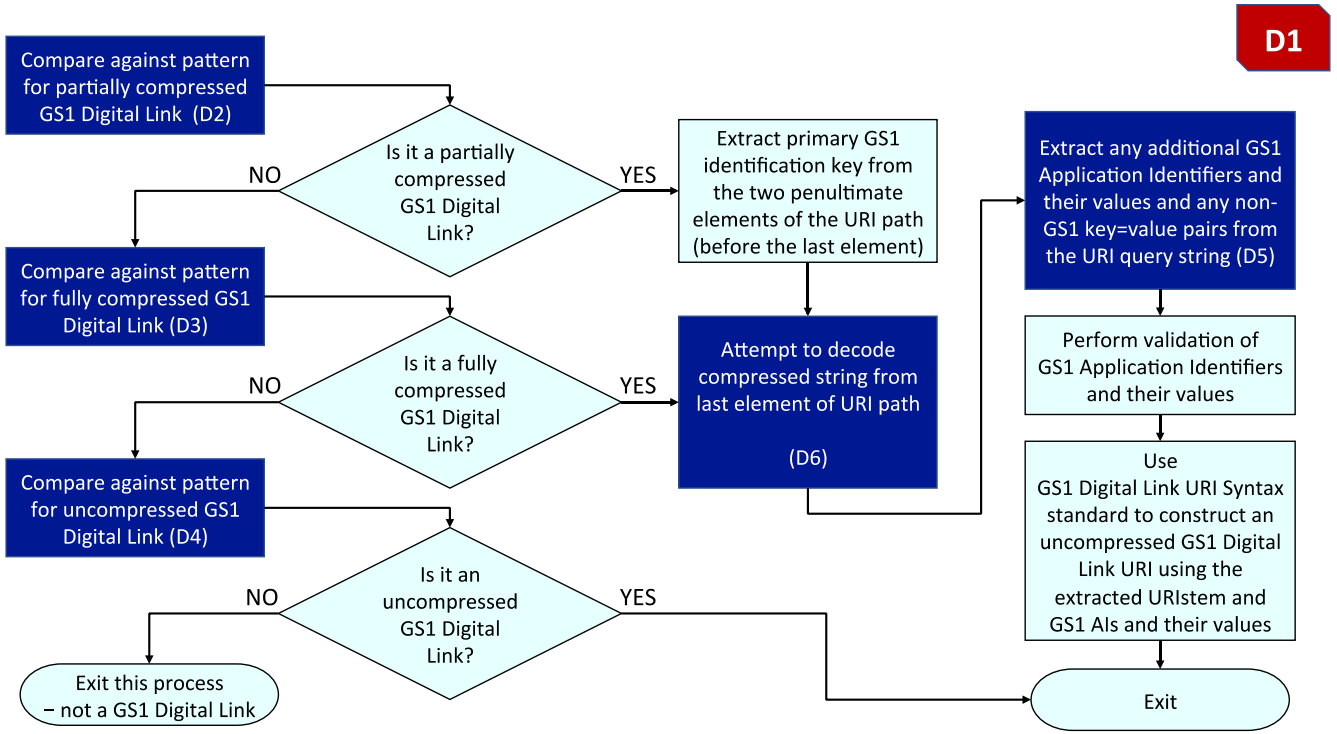


Flowchart C23 is the final high-level flowchart following on from the initial high-level Flowchart C5, which explains how to format the compressed (or partially compressed) GS1 Digital Link URI. The binary string buffer is right-padded with '0' bits to reach a total of bits that is a multiple of 6. Each group of 6 bits is then converted back to a character using the URI-safe base 64 alphabet. This appears as the final element of the URI path information (which already contains the uncompressed primary key and its value in the case of a partially compressed GS1 Digital Link URI, as explained in Flowchart C5). Any remaining uncompressed key:value pairs should appear in the URI query string.

3.9 Decompression procedure and flowcharts

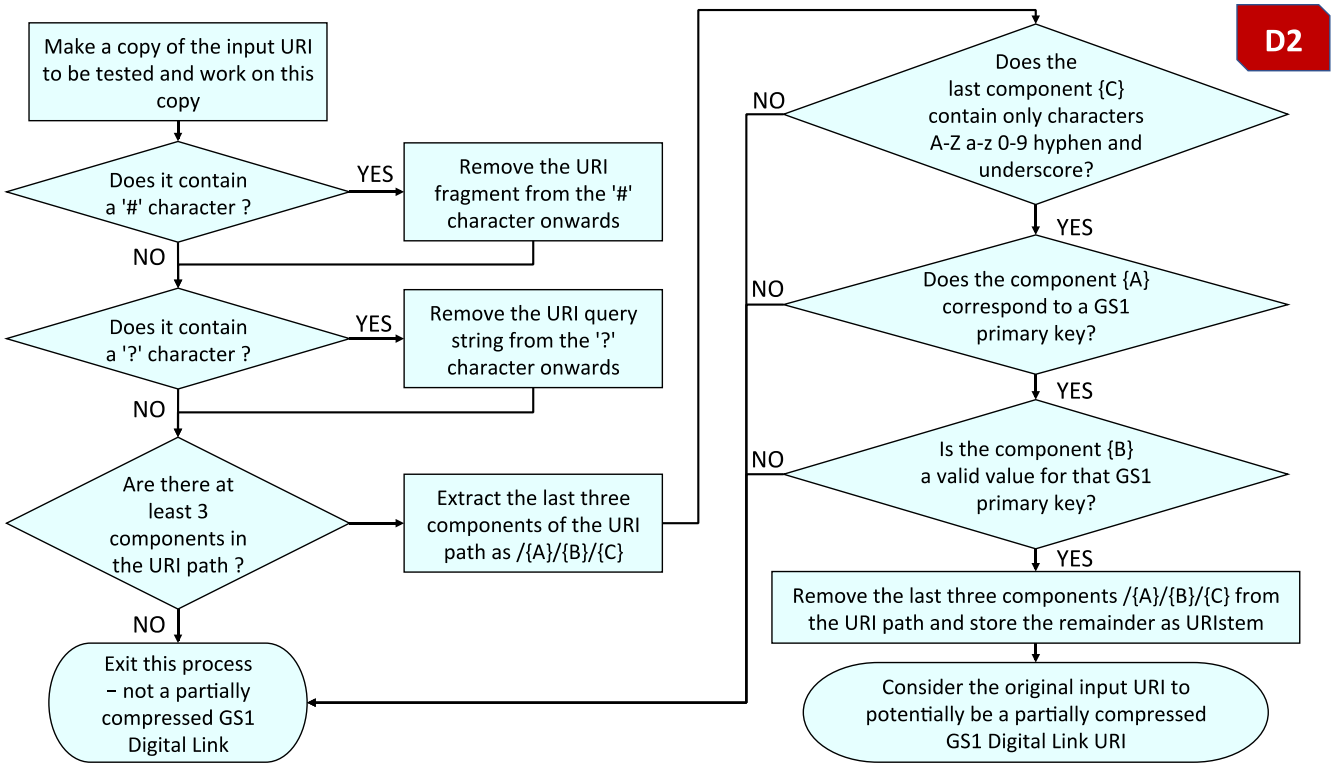
This section provides a set of flowcharts to describe the decompression procedure for fully or partially compressed GS1 Digital Link URIs. The result is an associative array of key:value pairs that can be translated to an uncompressed GS1 Digital Link URI.

Figure 3-24 D1: Top-level decompression flowchart



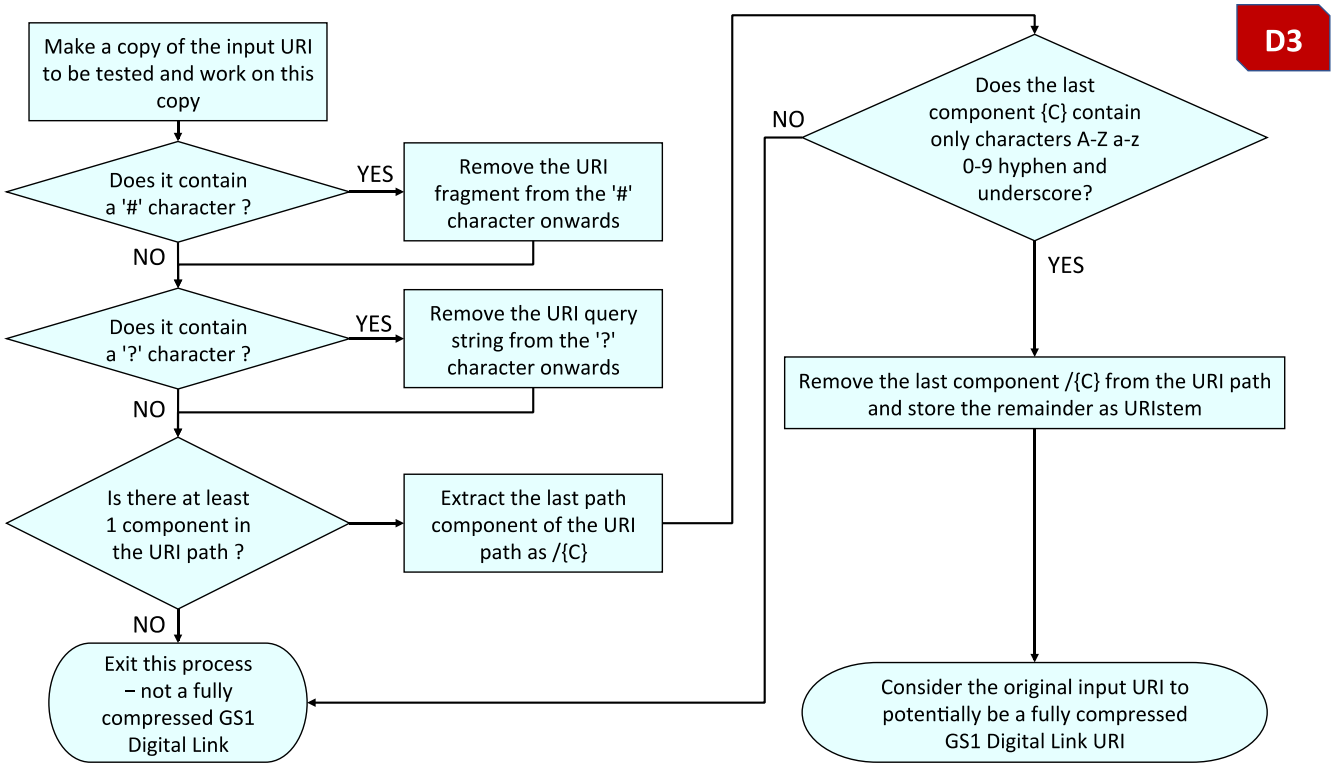
The decompression procedure begins by determining which kind of GS1 Digital Link URI is involved – uncompressed, partially compressed or fully compressed. Flowcharts D2, D3 and D4 are used to compare against patterns for partially compressed, fully compressed or uncompressed GS1 Digital Link URIs. For a partially or fully compressed GS1 Digital Link URI, flowchart D6 is used to attempt to decode information from the compressed string appearing as the last element of the URI path. This compressed string is expanded to a binary string in which the data is encoded efficiently, using the minimum number of bits depending on the type of value. Flowchart D5 is referenced, to extract any GS1 Applicatiomn Identifiers and their values and any non-GS1 key=value pairs from the URI query string.

Figure 3-25 D2: Compare against pattern for partially compressed GS1 Digital Link URI



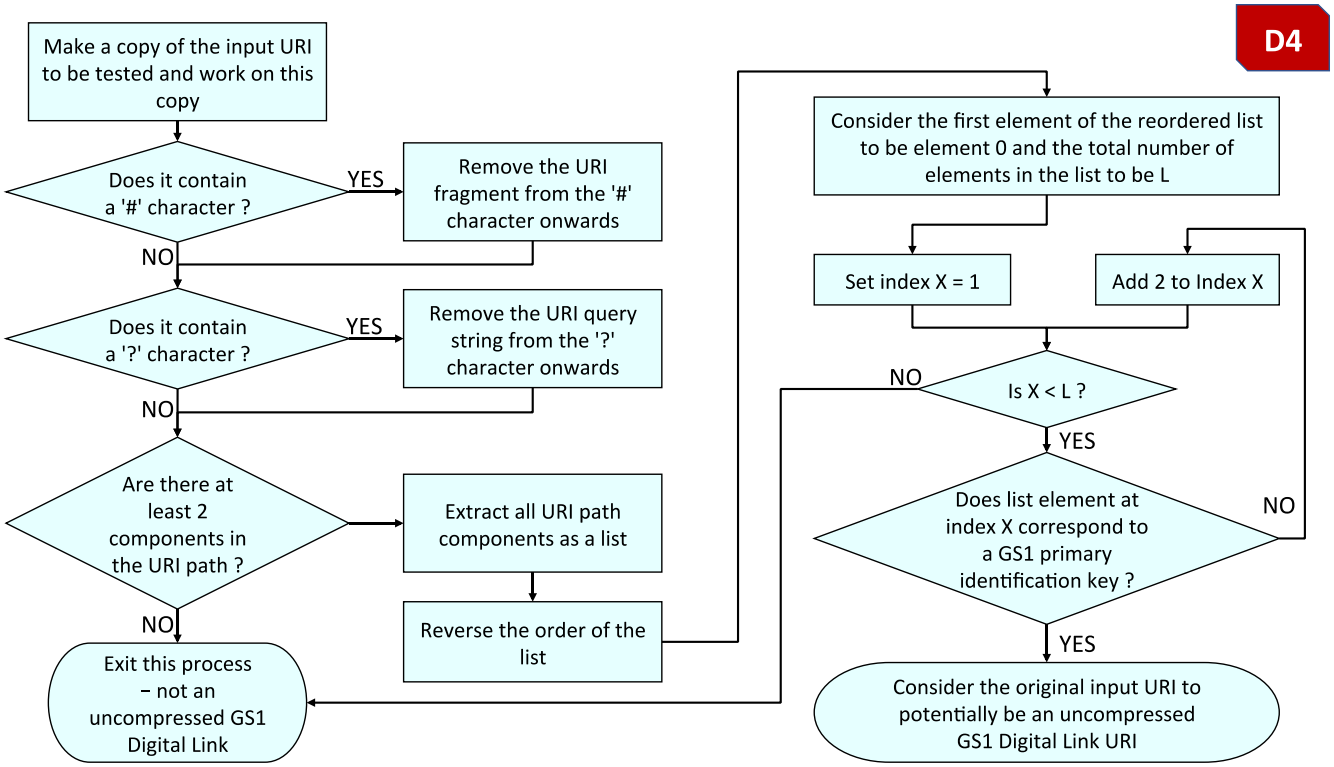
Flowchart D2 explains how to check for a partially compressed GS1 Digital Link URI in which the last component of the URI path consists of only characters from the URI-safe base 64 alphabet (A-Z a-z 0-9 hyphen and underscore) and is preceded by two URI path components, the first of which must correspond to a primary GS1 identification key such as GTIN, expressed either using numeric GS1 Application Identifiers or using the alphabetic short names defined in the GS1 Digital Link URI syntax standard [DL-URI].

Figure 3-5 D3: Compare against pattern for fully compressed GS1 Digital Link URI



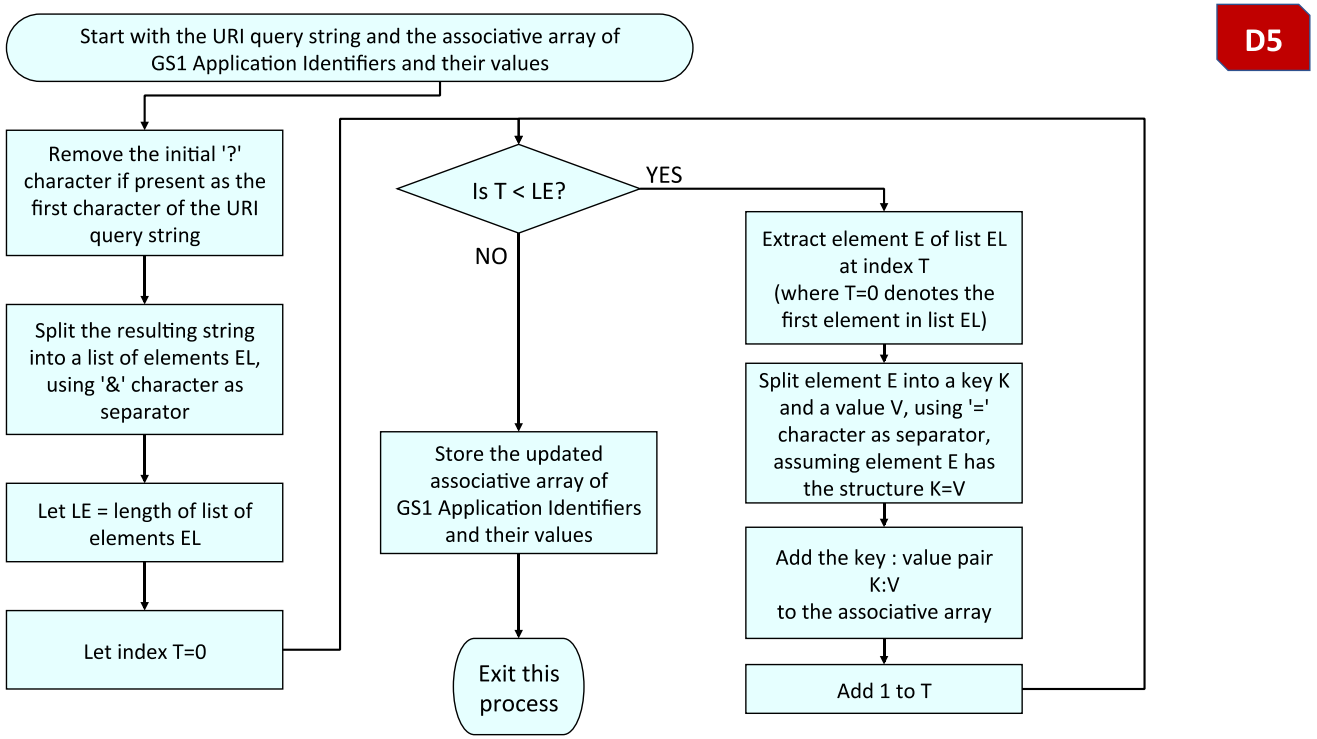
Flowchart D3 explains how to check for a partially compressed GS1 Digital Link URI in which the last component of the URI path consists of only characters from the URI-safe base 64 alphabet (A-Z a-z 0-9 hyphen and underscore). Note that because this pattern would also match a partially compressed GS1 Digital Link URI, the test for a partially compressed GS1 Digital Link URI (using Flowchart D2) must be performed first.

Figure 3-6 D4: Compare against pattern for uncompressed GS1 Digital Link URI



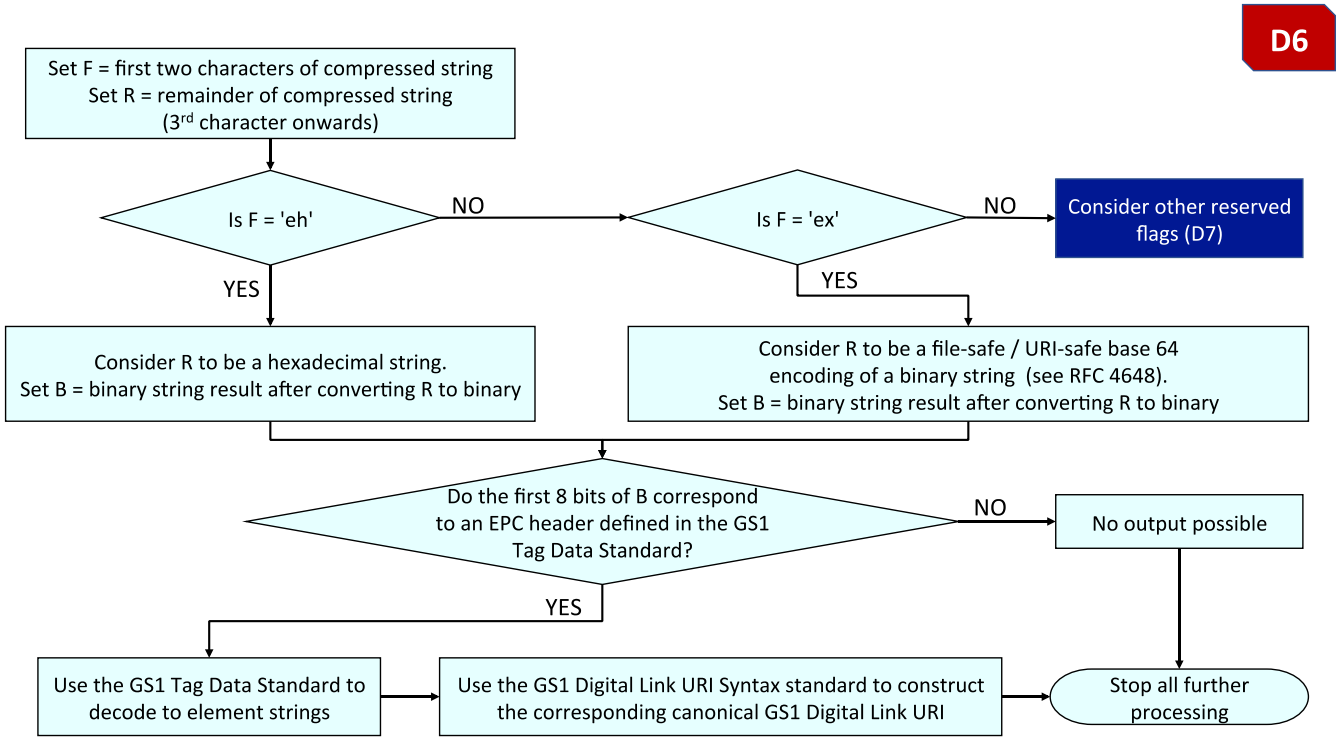
Flowchart D4 explains how to check for an uncompressed GS1 Digital Link URI. The URI path information can be analysed from right to left, considering two components at a time, testing whether the first of the pair of components corresponds to a primary GS1 identification key such as GTIN. Note that this pattern will not match a partially compressed GS1 Digital Link URI because the final URI path component of a partially compressed GS1 Digital Link URI is a compressed string and the primary GS1 identification key would appear two components to the left of that final component, whereas for an uncompressed GS1 Digital Link URI, the primary GS1 identification key must appear an odd number of components before the final component of the URI path information. In this way, the two patterns are mutually exclusive.

Figure 3-7 D5: Extraction of GS1 Application Identifiers and their values from the URI query string



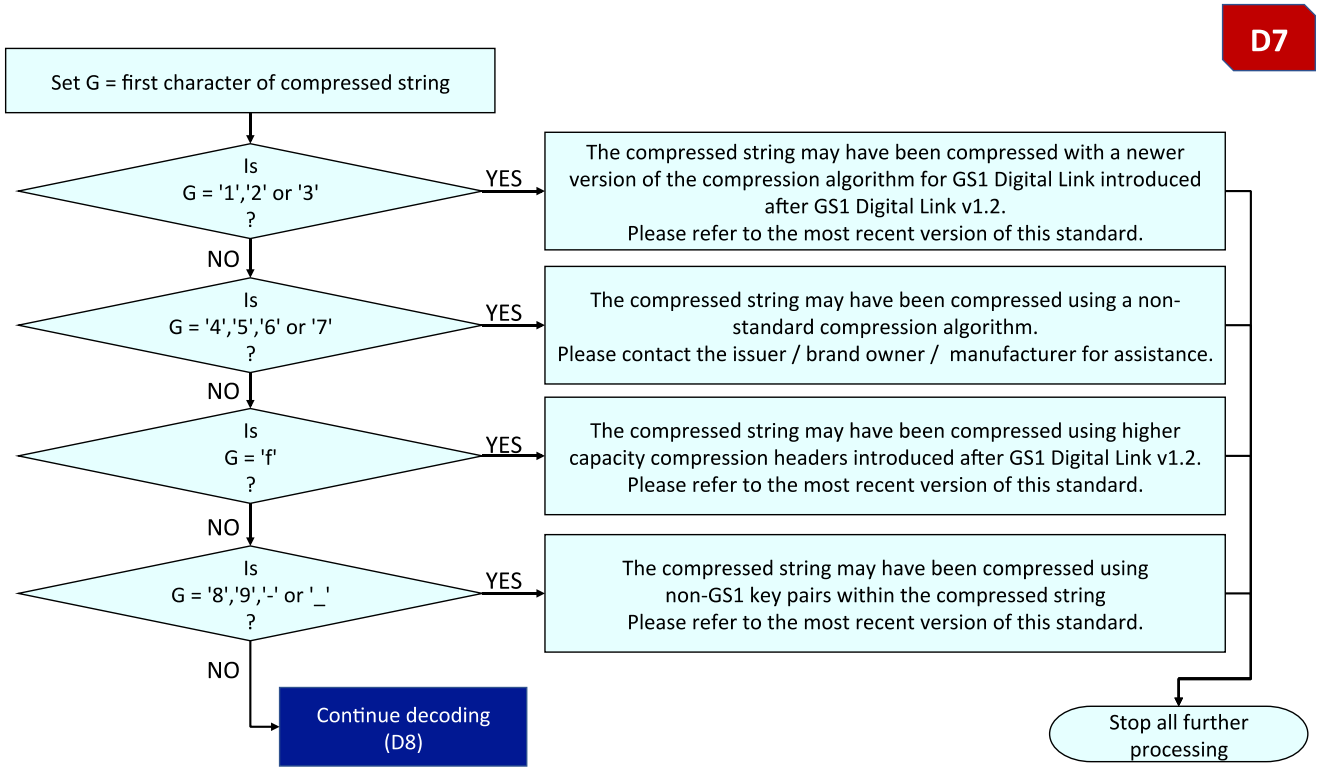
Flowchart D5 explains how to extract GS1 Application Identifiers and their values from the URI query string

Figure 3-8 D6: Check whether the compressed string expresses an EPC binary string



Flowchart D6 includes tests to check whether the initial two characters of the compressed string are 'eh' or 'ex'. 'eh' indicates that what follows from the third character of the compressed string is a hexadecimal representation of an EPC binary string. 'ex' indicates that what follows from the third character of the compressed string is an encoding of the EPC binary string using the file-safe / URI-safe base 64 alphabet defined in RFC 4648. If an EPC binary string can be extracted, the GS1 Tag Data Standard must be used to decode to GS1 element string format, then the GS1 Digital Link URI syntax must be used to construct the corresponding canonical GS1 Digital Link URI. If the first two characters of the compressed string are neither 'eh' nor 'ex', then processing continues to Flowchart D7.

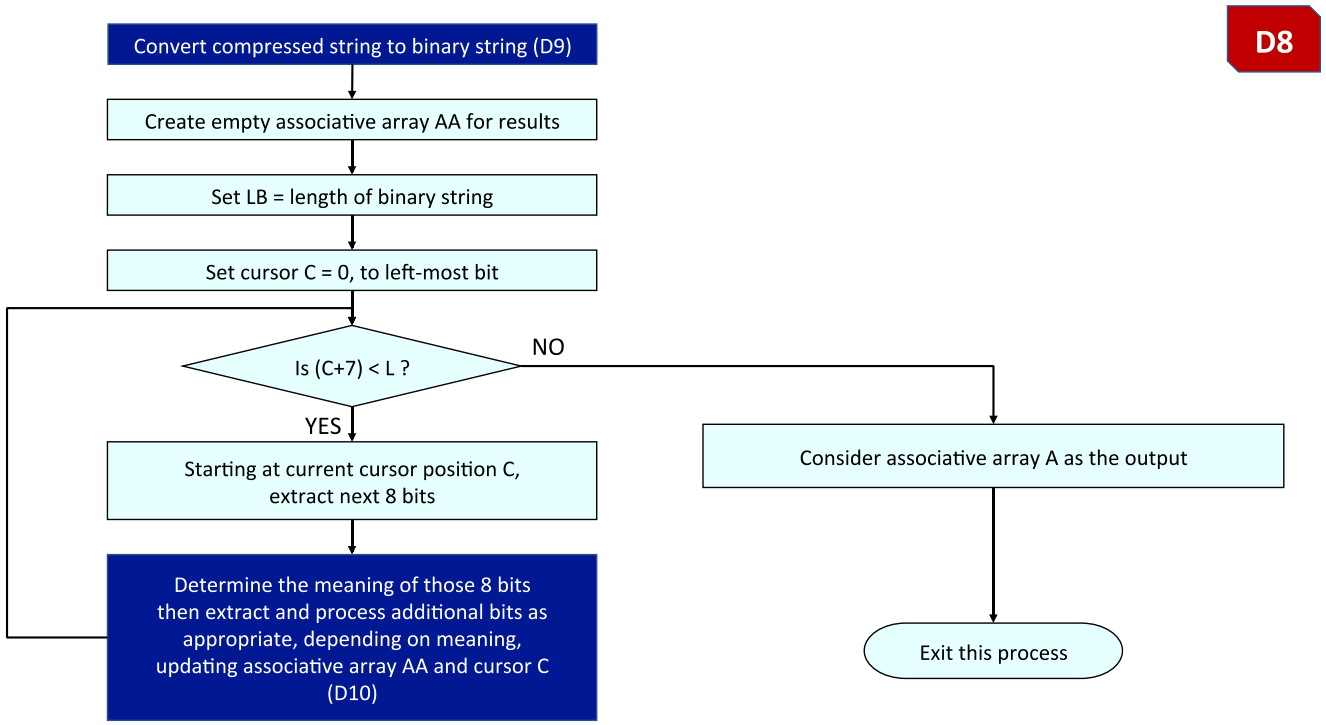
Figure 3-30 D7: Check whether the compressed string uses reserved compression headers



Flowchart D7 includes tests for reserved compression headers by inspection of the initial character of the compressed string.

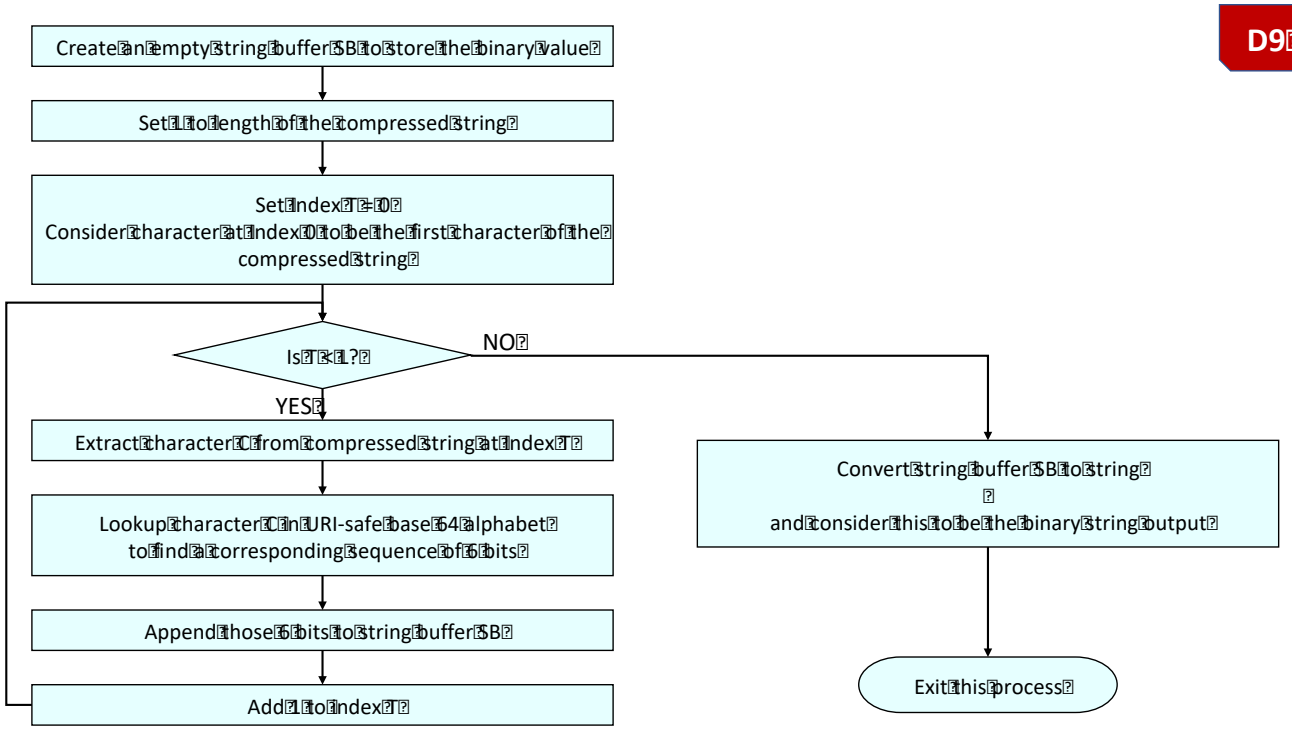
Further processing continues in Flowchart D8 if none of the reserved compression headers are used.

Figure 3-31 D8: High-level decoding flowchart, references flowcharts D9 (convert URI-safe base 64 characters to binary) and D10 (extract meaning and populate associative array)



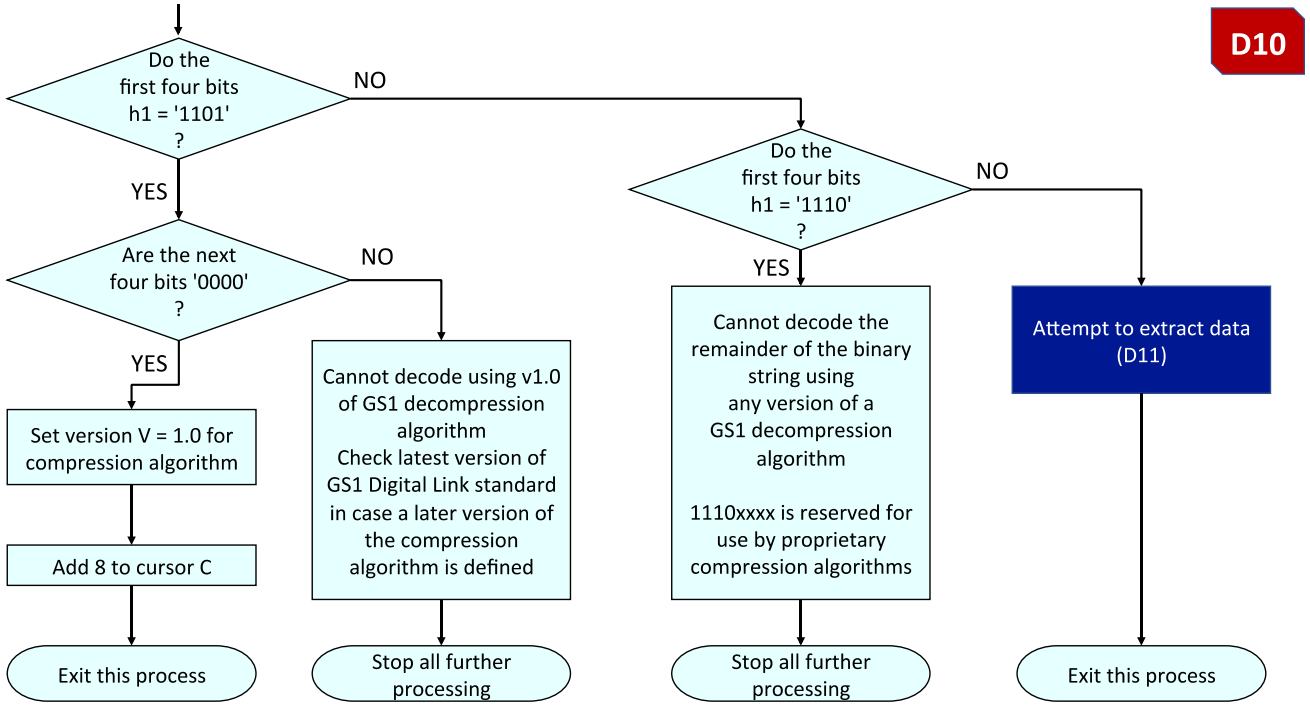
Flowchart D8 is the high-level flowchart for decoding the compressed string that is the final URI path component of a partially or fully compressed GS1 Digital Link URI. It references Flowchart D9 for the conversion from the URI-safe base 64 alphabet characters into a binary string. The main processing loop of Flowchart D8 begins by extracting 8 bits (provided that there are at least 8 bits remaining in the binary string) and then references Flowchart D10 and its dependent flowcharts to determine the meaning of those 8 bits.

Figure 3-32 D9: Convert the compressed string from URI-safe base 64 characters to a binary string



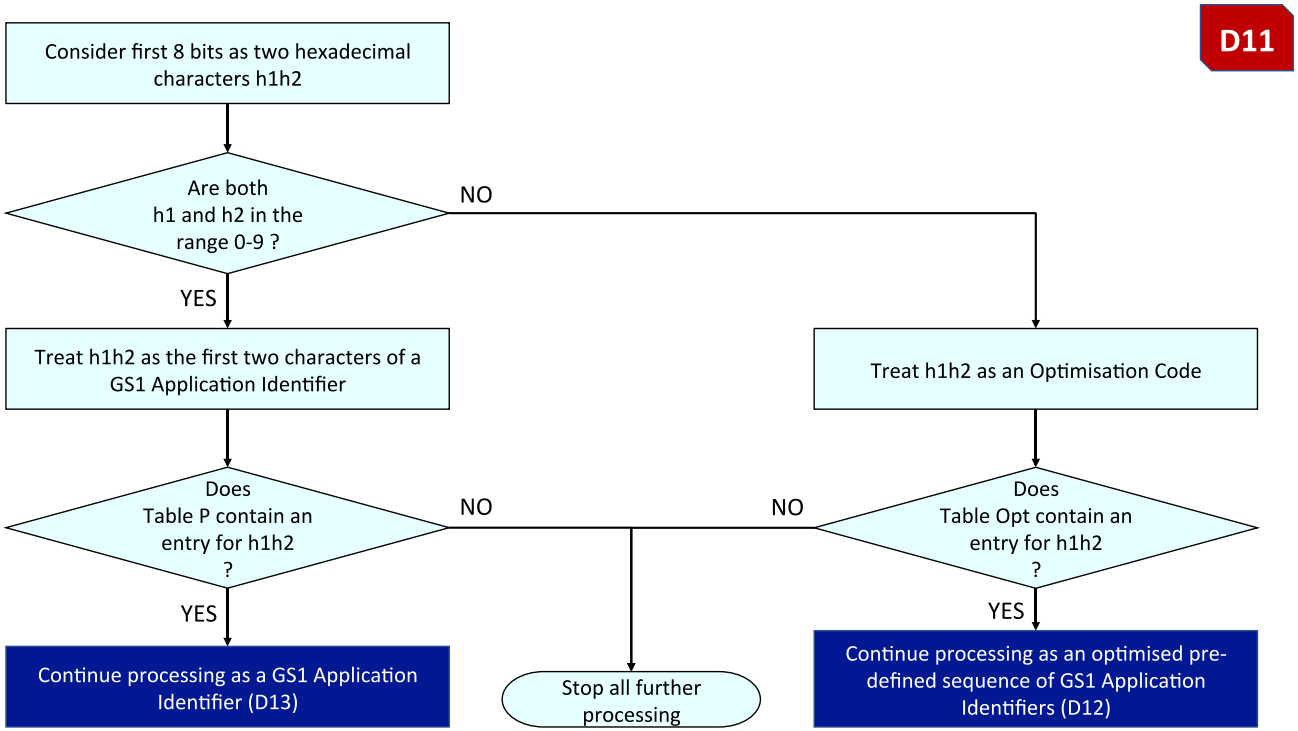
Flowchart D9 explains how to convert the compressed string expressed using URI-safe base 64 characters into a binary string. Each character encodes a sequence of 6 bits. The compressed string is decoded from left to right and the corresponding set of 6 bits per URI-safe base 64 character are also appended to a string buffer for the binary string, also from left to right. Note also that the binary string will be processed from left to right, rather than being treated as a very large binary integer. This means that any leading zeros at the left of the binary string are significant.

Figure 3-33 D10: Check the meaning of an 8-bit sequence. This includes checking whether a standard version is indicated by 1101xxxx (Dx) – and whether version 1.0 (1101 0000 = D0), otherwise check if a proprietary algorithm (1110 xxxx = Ex) is being used, otherwise attempt to decode data, referring to flowchart D11



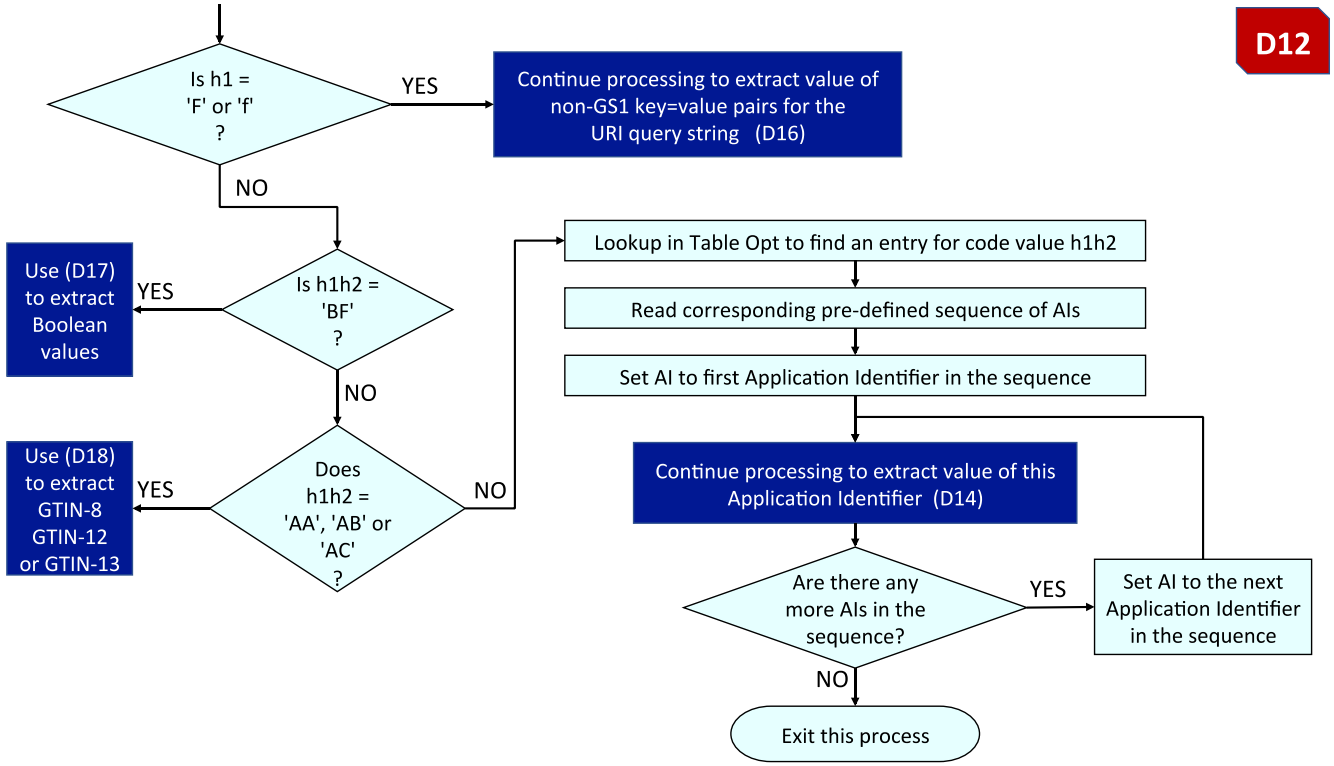
Flowchart D10 explains a sequence of checks on each 8-bit sequence read from the binary string on each iteration of the main loop of Flowchart D8. This includes detection of a potential version number. Currently only version 1.0 of the GS1 decompression algorithm is defined and identified as binary sequence 11010000 (D0 in hexadecimal). If a pattern of 1110xxxx is encountered, this indicates a handover / extension point to a non-standard decompression algorithm and the GS1 decompression algorithm stops all further processing at this point. Otherwise, Flowchart D10 references D11 to attempt to extract data from the 8 bits. The 8 bits might correspond to the first two digits of a GS1 Application Identifier of 2,3 or 4 digits or it may correspond to an optimisation code corresponding to a pre-defined sequence of GS1 Application Identifiers.

Figure 3-34 D11: Extract data from binary string



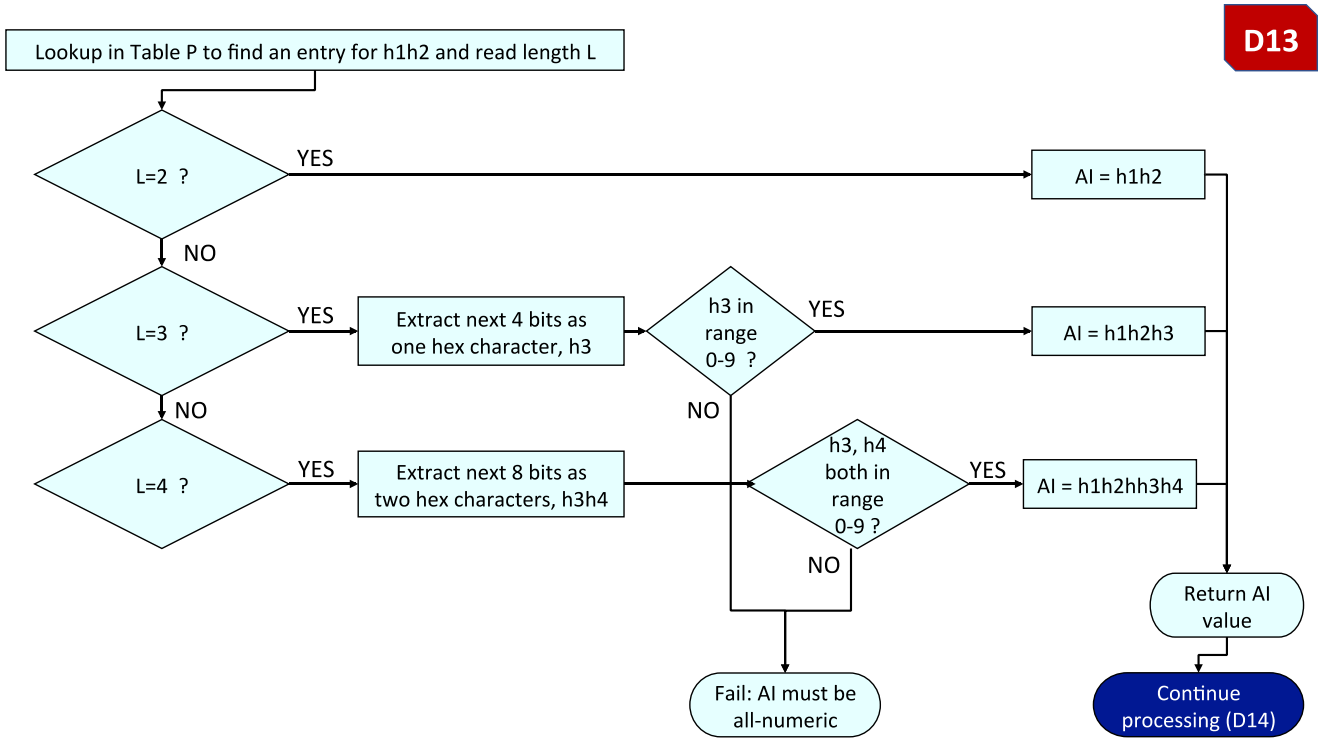
Flowchart D11 explains how to check whether the sequence of 8 bits corresponds to the first two digits of a GS1 Application Identifier of 2, 3 or 4 digits or whether it corresponds to an 8-bit optimisation code that corresponds to a predefined sequence of GS1 Application Identifiers, for more efficient compression of frequently-combined element strings. Treating the 8 bits as two hexadecimal characters h1h2 (each character corresponding to 4 bits), if both characters are in the range 0-9, then processing continues to Flowchart D13 provided that Table P includes an entry for digits h1h2. If either of h1h2 are outside 0-9 (i.e. at least one of them includes hex character a-f), h1h2 is considered as an optimisation code. If an entry for h1h2 can be found in Table Opt (table of optimisation codes), processing continues to Flowchart D12 for handling of optimisations consisting of pre-defined sequences of GS1 Application Identifiers.

Figure 3-35 D12: Processing as an optimised pre-defined sequence of GS1 Application Identifiers



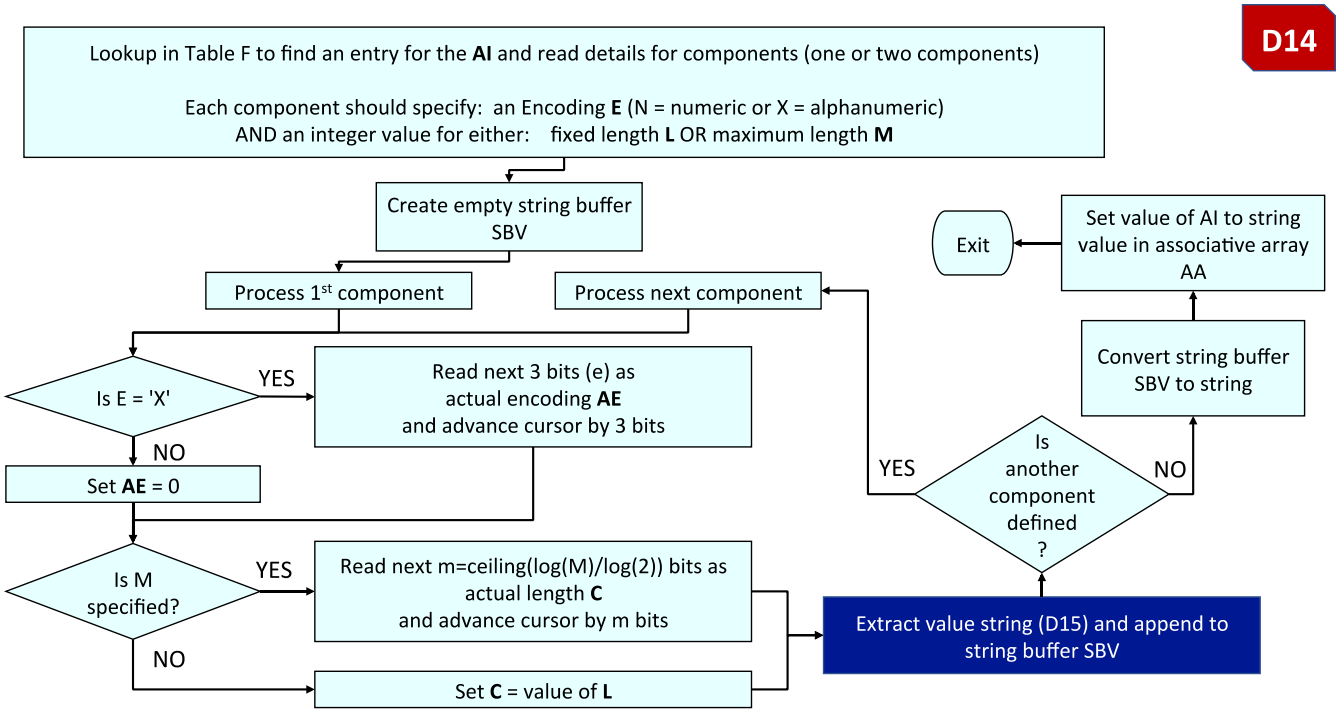
Flowchart D12 begins by performing a test whether h1 corresponds to hex character f/F (1111 in binary). If so, processing continues to Flowchart D16 because the optimisation code range Fx (1111xxxx) is reserved to support compression of non-GS1 key:value pairs from the uncompressed URI query string into the compression string. If h1h2 is "BF", then Flowchart D17 is used to extract Boolean values for GS1 Application Identifiers (4321), (4322) or (4323). If h1h2 is "AA", "AB" or "AC", Flowchart D18 is used to extract values of a GTIN-8, GTIN-12 or GTIN-13 respectively. Otherwise, the corresponding entry for h1h2 in Table Opt is read, to obtain the pre-defined sequence of GS1 Application Identifiers. Each of these is processed in turn, in the loop of Flowchart D12, referencing Flowchart D14 for extraction of the value for each GS1 Application Identifier.

Figure 3-36 D13: Process as a single GS1 Application Identifier



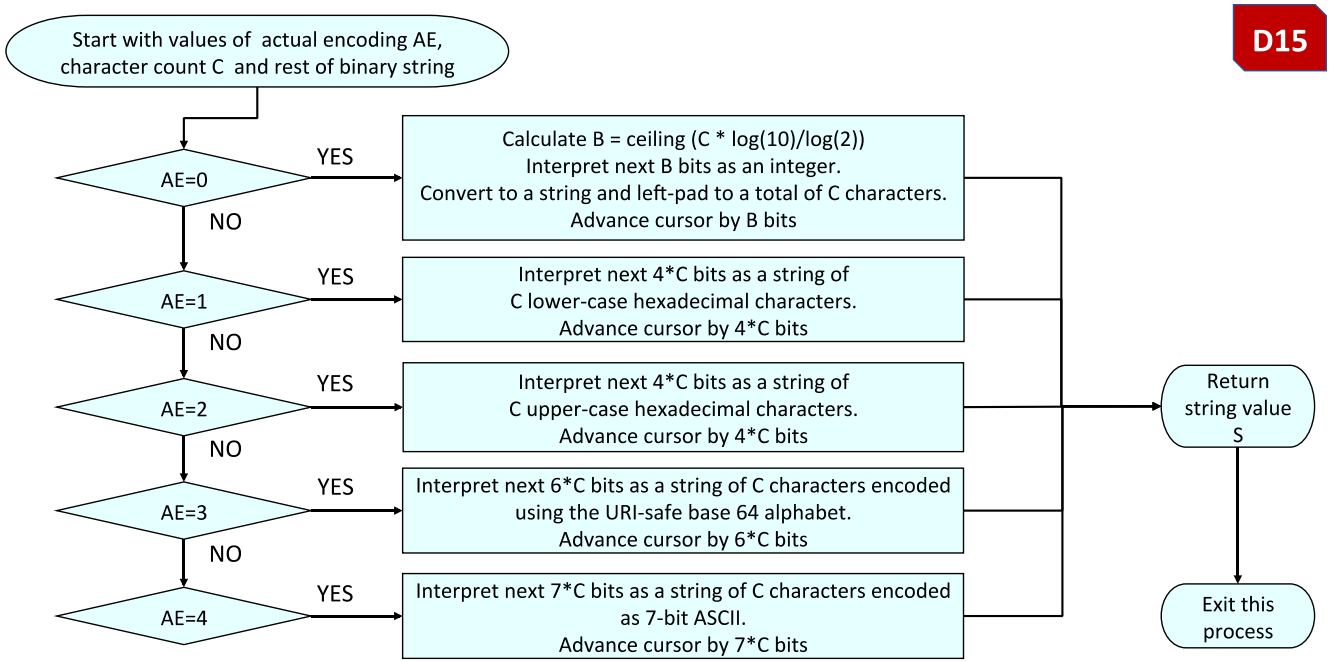
Flowchart D13 is used to determine whether the initial digits h1h2 correspond to a 2-digit, 3-digit or 4-digit GS1 Application Identifier. Table P contains the current rules for making this determination based on the initial two digits. If Table P indicates a 3-digit GS1 Application Identifier (e.g. 414), a further four bits are read from the binary string as hex character h3, resulting in GS1 Application Identifier h1h2h3. If Table P indicates a 4-digit GS1 Application Identifier (e.g. 8004), a further 8 bits are read from the binary string as two hex characters, h3h4, resulting in GS1 Application Identifier h1h2h3h4. Further processing then continues to Flowchart D14 to extract the value for the GS1 Application Identifier key identified in this flowchart.

Figure 3-37 D14: Determine values for actual encoding AE and character count C for each component



Flowchart D14 explains how to extract the value for each GS1 Application Identifier. Firstly, a lookup in Table F seeks an entry for the GS1 Application Identifier and reads details for one or two components. Each component specifies an encoding E (N for numeric, X for alphanumeric) and an integer value for either L (fixed length) or M (maximum length). If encoding E = X, a 3-bit encoding indicator is extracted from the binary string and converted to integer AE that expresses the actual encoding. Otherwise, AE is set to zero for numeric encoding. If a value is specified for the maximum length, M, a length indicator is read, using an appropriate number of m bits (see formula for m). These are converted to decimal to determine the actual length of a component that was permitted to be variable length. Flowchart D15 is referenced for the extraction of the actual value for that component, which is appended to a string buffer. If Table F indicated a second component, this is also processed. The final value of the string buffer is associated with the GS1 Application Identifier under consideration.

Figure 3-9 D15: Extract the value from the binary string



Flowchart D15 explains how to extract an appropriate number of bits for a specific value of actual encoding AE and character count C (determined in Flowchart D14). This supports numeric encoding as a binary integer (at ≈ 3.32 bits per digit), lower-case and upper-case hexadecimal string (both at 4 bits per character), URI-safe base 64 strings (at 6 bits per character) and finally ASCII (at 7 bits per character).

Figure 3-39 D16: Extract the value for key=value pairs

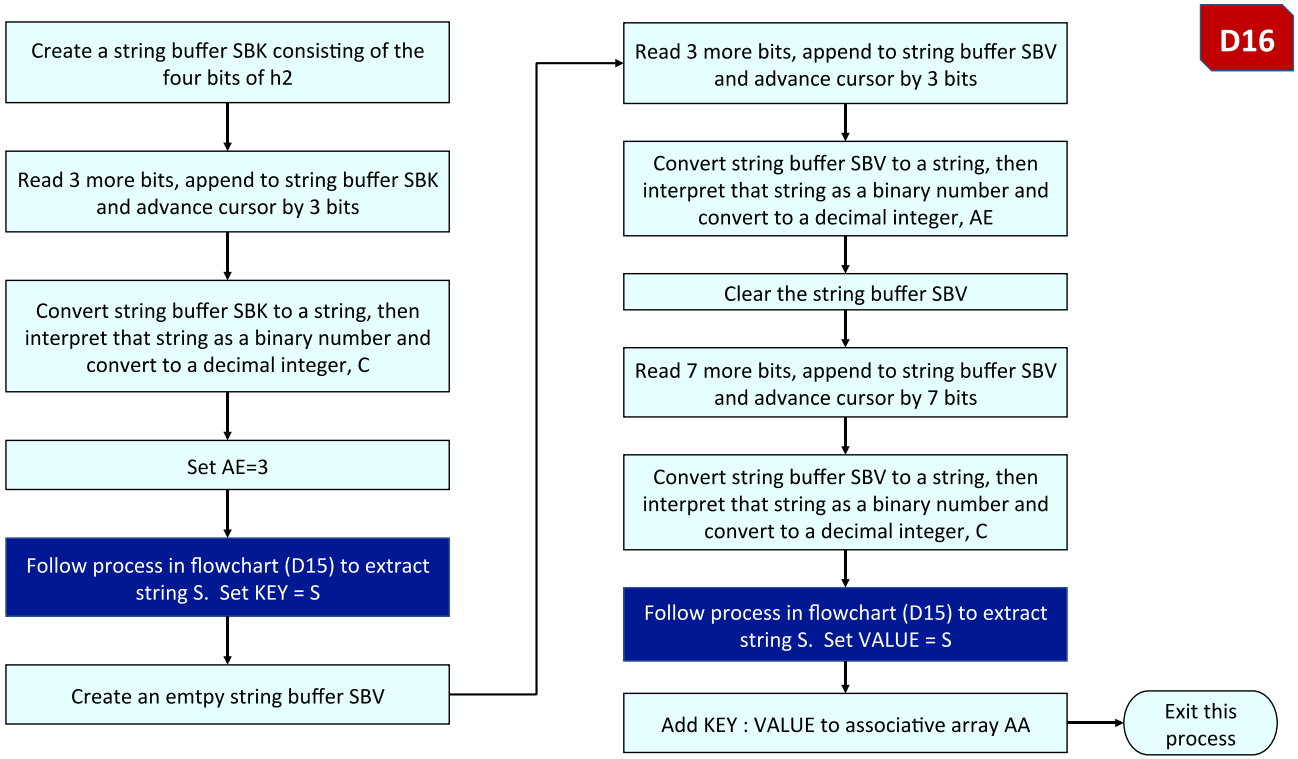
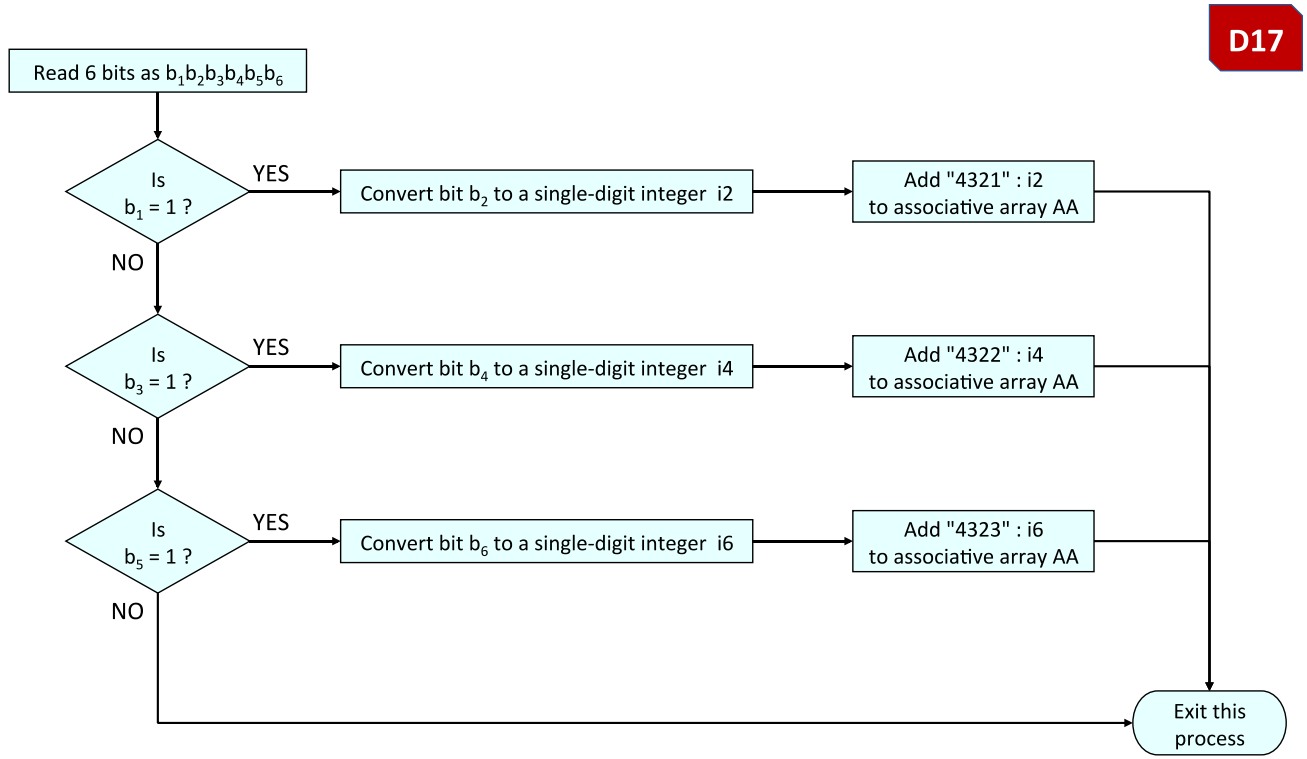


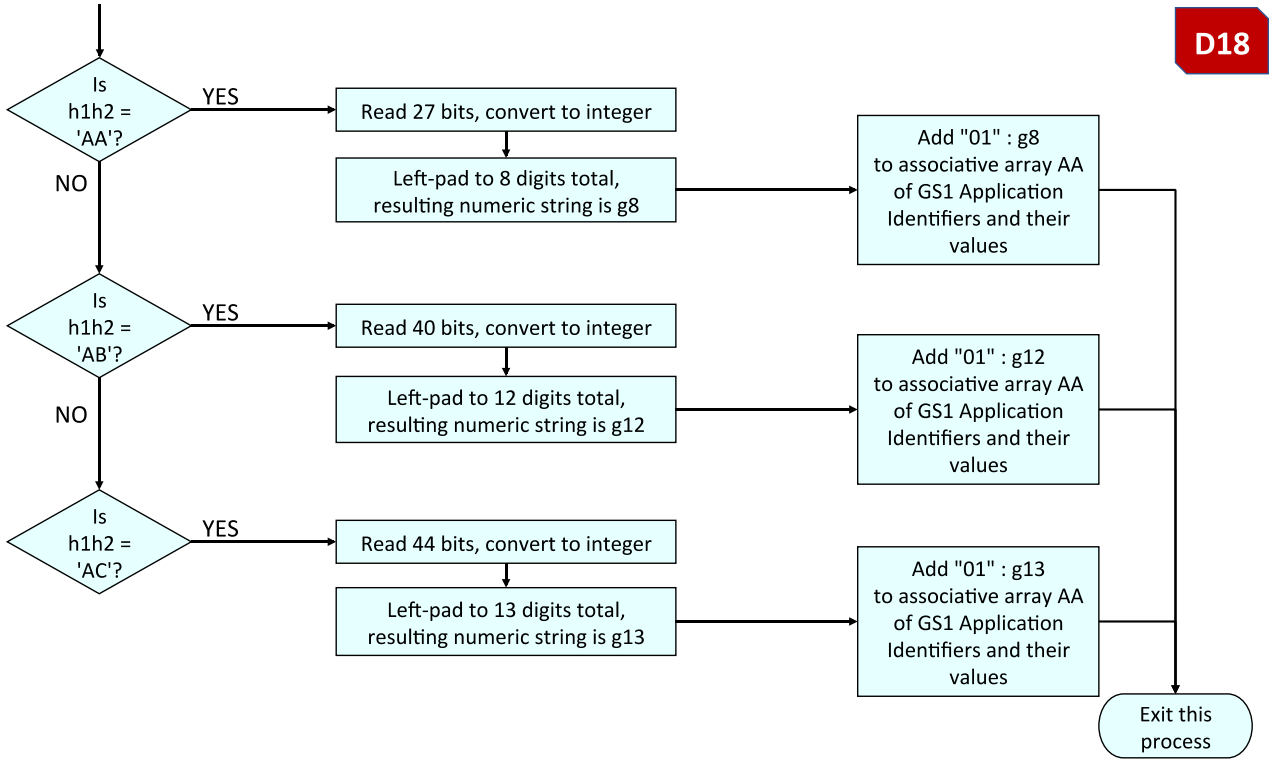
Figure D16 explains how to extract non-GS1 key:value pairs from a compressed string; after decompression these are restored to the URI query string. The four bits of h2 and a further 3 bits are interpreted as a 7-bit length indicator, permitting a key up to 127 characters. The key is always encoded using the URI-safe base 64 alphabet at 6 bits per character. An empty string buffer is created for storing the value. A 3-bit encoding indicator AE and a 7-bit length indicator C are read from the binary string, then flowchart D15 is used to extract the value, depending on the value of AE and C. The key:value pair are then added to the associative array AA that also includes key:value pairs for GS1 element strings in which the key is a GS1 Application Identifier.. The associative array AA is converted back to an uncompressed GS1 Digital Link URI using translation methods. Any non-GS1 key:value pairs appear in the URI query string.

Figure 3-40 D17: Flowchart for extraction of Boolean values for GS1 Application Identifiers (4321), (4322) and (4323)



Flowchart D17 explains how to decompress the 6-bit field that follows compression header "BF", to extract Boolean values (0=false, 1=true) for GS1 Application Identifiers (4321), (4322) or (4323). Further explanation can be found in section 3.5.2.

Figure 3-41 D18: Flowchart for extraction of GTIN-8, GTIN-12 or GTIN-13 with preservation of their length



Flowchart D18 explains how to decompress any of the following:

- a binary field of 27 bits that follows compression header "AA" as a GTIN-8 value
- a binary field of 40 bits that follows compression header "AB" as a GTIN-12 value
- a binary field of 44 bits that follows compression header "AC" as a GTIN-13 value

4 Glossary

The glossary lists the terms and definitions that are applied in this document. Please refer to the *www.gs1.org/glossary* for the online version.

Term	Definition
Attribute	An element string that provides additional information about an entity identified with a GS1 identification key, such as batch number associated with a Global Trade Item Number (GTIN).
Brand Owner	The organisation that owns the specifications of a trade item, regardless of where and by whom it is manufactured. The brand owner is normally responsible for the management of the Global Trade Item Number (GTIN).
Consumer	Often considered as the "recipient" of the supply chain in the past, today's consumer is an active part of the supply chain and expects more data, with higher accuracy, and greater ease.
Consumer Product Variant (CPV)	An alphanumeric attribute of a GTIN assigned to a retail consumer trade item variant for its lifetime.
Domain name	A domain name is an identification string that defines a realm of administrative autonomy, authority or control within the Internet. Domain names are formed by the rules and procedures of the Domain Name System (DNS). Any name registered in the DNS is a domain name. Domain names are used in various networking contexts and application-specific naming and addressing purposes. Domain names provide a abstraction layer that separates a registered name for an organisation or activity from the actual internet addresses (IP addresses) that provide its associated information services such as its Website, its e-mail server etc. The system that connects the domain names with the corresponding IP addresses is the Domain Name System (DNS).
Element string	The combination of a GS1 Application Identifier and GS1 Application Identifier data field.
GS1 Application identifier	The field of two or more digits at the beginning of an element string that uniquely defines its format and meaning.
GS1 Application identifier data field	The data used in a business application defined by one GS1 Application Identifier.
GS1 Barcode	A data carrier which encodes GS1 Application Identifier element strings.
GS1 Barcode using GS1 Application Identifiers	All GS1 endorsed barcode symbologies that can encode more than a GTIN namely GS1-128, GS1 DataMatrix, GS1 DataBar and Composite and GS1 QR Code.
GS1 Identification key	A unique identifier for a class of objects (e.g. a trade item) or an instance of an object (e.g. a logistic unit).
GS1 key qualifier	A key qualifier is an additional attribute that is designated for use as part of a compound key (e.g., GTIN + serial number is a compound key, with the serial number being a key qualifier for the GTIN)
GS1 Resolver	A Web server that is able to understand the GS1 Digital Link URI syntax
GS1 Digital Link URI	A Web URI conforming to the GS1 Digital Link URI syntax.
HR14	HttpRange14Webography. The chronology of a permathread, Sandro Hawke, W3C Wiki, 2003 https://www.w3.org/wiki/HttpRange14Webography
HTTP status codes	The status-code element is a three-digit integer code giving the result of the attempt to understand and satisfy the request.
Identification number	A numeric or alphanumeric field intended to enable the recognition of one entity versus another.
LGTIN (GTIN + Lot/Batch)	A compound key formed from the combination of GTIN [AI (01)] and Batch/Lot identifier [AI (10)]. LGTIN is defined as an EPC Class URN in the current GS1 Tag Data Standard (v1.11), sections 6.4.1 and 7.14, which describes the mapping between the EPC Class URN format for LGTIN and the corresponding element string.

Term	Definition
Media Type (also known as MIME type or Content type)	A two-part string identifier that indicates a data format as a pair of type and subtype, e.g. image/jpeg, image/gif, image/png, text/html, text/rtf Media types are sometimes also referred to as MIME types (MIME is an acronym of Multipurpose Internet Mail Extensions) or Content Types (after the HTTP header that indicates the Media type)
Mobile scanning	An approach to giving consumers access to additional information or services about trade items through their mobile device. It is the ability to retrieve additional information about the trade item through mobile devices or in general between link a trade item with virtual information or services.
Parsing	The process of analysing the structure of a sentence or URI structure in order to extract relevant information from it. Note that within the context of EPC URN structures, parsing refers to the ability to extract structural components within the EPC structure, e.g. for the purpose of matching against EPC URN patterns.
QR Code®	A two-dimensional matrix symbology consisting of square modules arranged in a square pattern. The symbology is characterised by a unique finder pattern located at three corners of the symbol. QR Code® symbols are read by two-dimensional imaging scanners or vision systems
Reference GS1 Digital Link URI	A GS1 Digital Link URI that uses the id.gs1.org domain
Resolver	A resolver connects a GS1-identified item to one or more online resources that are directly related to it. The item may be identified at any level of granularity, and the resources may be either human or machine readable. Examples include product information pages, instruction manuals, patient leaflets and clinical data, product data, service APIs, marketing experiences and more.
Retailer	An organisation engaged in the sale and distribution of products to consumers. Also includes online retailers / e-tailers
URI	Uniform Resource Identifier. A string of characters used to identify a resource. The resource may be an information resource such as a Web page or a thing in the real world, such as a physical object, person or location. URIs refer to the superset of Uniform Resource Names (URNs), Uniform Resource Locators (URLs) and Web URIs (which can function both as globally unambiguous names, while also behaving like URLs by enabling intuitive retrieval of related information via the Web).
URI path information	A path consists of a sequence of path segments separated by a slash ("/") character. A path is always defined for a URI, though the defined path may be empty (zero length). The path component contains data, usually organized in hierarchical form, that, along with data in the non-hierarchical query component, serves to identify a resource within the scope of the URI's scheme and naming authority (if any). The path is terminated by the first question mark ("?") or number sign ("#") character, or by the end of the URI.
URI query string	The query component contains non-hierarchical data that, along with data in the path component, serves to identify a resource within the scope of the URI's scheme and naming authority (if any). The query component is indicated by the first question mark ("?") character and terminated by a number sign ("#") character or by the end of the URI.
URL	Uniform Resource Locator (URL), a specific type of URI colloquially known as Web address. A URL is a URI starting with http or https .

5 References

[DL1]

GS1 Digital link version 1.0 Originally titled GS1 Web URI Structure. Mark Harrison, Phil Archer, Dom Guinard et al. GS1 Ratified Standard, August 2018 *https://www.gs1.org/standards/Digital-Link/1-0*

[DL 1.1]

GS1 Digital Link version 1.1 Mark Harrison, Phil Archer et al GS1 Ratified Standard, February 2020

https://www.gs1.org/docs/Digital-Link/GS1_Digital_link_Standard_i1.1.pdf

[DL-Resolution]
GS1 Digital Link: Resolution. Phil Archer, Mark Harrison, Dom Guinard et al. @@@Date@@@, GS1 ratified standard @@@URL@@@

[DL-Semantics]
GS1 Digital Link: Semantics, Mark Harrison, Phil Archer et al. @@@Date@@@, GS1 ratified standard @@@URL@@@

[DL-URI]
GS1 Digital Link: URI Syntax. Mark Harrison, @@@Date@@@, GS1 ratified standard @@@url@@@

[GENSPECS]

GS1 General Specifications V20.0. GS1 Ratified Standard January 2020

https://www.gs1.org/sites/default/files/docs/barcodes/GS1_General_Specifications.pdf

[RFC 2606]

Reserved Top Level Domain Names D Eastlake, A Panitz. IETF June 1999 https://tools.ietf.org/html/rfc2606

[RFC 4648]

The Base16, Base32, and Base64 Data Encodings. S Josefson. IETF October 2006 *https://www.rfc-editor.org/rfc/rfc4648.txt*

[RFC 6761]

Special-Use Domain Names. S Cheshire, M Krochmal. IETF February 2013 *https://tools.ietf.org/html/rfc6761*

[TDS]

Tag Data Standard GS1 ratified standard, version 1.13, November 2019

https://www.gs1.org/standards/epcrfid-epcis-id-keys/epc-rfid-tds/1-13

[TDT]

Tag Data Translation standard, version 1.6. GS1 ratified standard, 12 October 2011. https://www.gs1.org/standards/epcrfid-epcis-id-keys/epc-rfid-tds/1-13

6 Change log

Changes since version 1.1 of this work are described in section 3.1

A.1 Intellectual Property

A.1.1 Introduction and Disclaimer

GS1®, under its IP Policy, seeks to avoid uncertainty regarding intellectual property claims by requiring the participants in the Work Group that developed the GS1 Digital Link: Compression Standard Release 1.2 (for the purpose of this paragraph A.1.1, the "**Standard**") to agree to grant to GS1 members a royalty-free licence or a RAND licence to Necessary Claims, as that term is defined in the GS1 IP Policy. Furthermore, attention is drawn to the possibility that an implementation of one or more features of the Standard may be the subject of a patent or other intellectual property right that does not involve a Necessary Claim. Any such patent or other intellectual property right is not subject to the licencing obligations of GS1. Moreover, the agreement to grant licences provided under the GS1 IP Policy does not include IP rights and any claims of third parties who were not participants in the Work Group.

Accordingly, GS1 recommends that any organisation developing an implementation designed to be in conformance with the Standard should determine whether there are any patents that may encompass a specific implementation that the organisation is developing in compliance with the Standard and whether a licence under a patent or other intellectual property right is needed. Such a determination of a need for licencing should be made in view of the details of the specific system designed by the organisation in consultation with their own patent counsel.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR PARTICULAR PURPOSE, OR ANY WARRANTY OTHER WISE ARISING OUT OF THIS SPECIFICATION. GS1 disclaims all liability for any damages arising from use or misuse of this document, whether special, indirect, consequential, or compensatory damages, and including liability for infringement of any intellectual property rights, relating to use of information in or reliance upon this document.

GS1 retains the right to make changes to this document at any time, without notice. GS1 makes no warranty for the use of this document and assumes no responsibility for any errors which may appear in the document, nor does it make a commitment to update the information contained herein.

GS1 and the GS1 logo are registered trademarks of GS1 AISBL.

A.1.2 Notices

A.1.2.1 Patents and Patent Applications of Mobilead SAS

MobiLead SAS of 1 Cour du Havre, 75008 Paris (France), the owner of the granted patent and five patent applications listed in this sub-section (for the purpose of this specific paragraph A.1.2.1, the "**Patent and patent applications**") who participated to the Work Group designing the GS1 Digital Link: Compression Standard Release 1.2, gave notice that they believe the Patent and patent applications contain Essential Claims for the implementation of the GS1 Digital Link: Compression Standard Release 1.2 as follows.

Patent US 9,864,889 and patent applications EP 3147890, US 20180204034 and EP 17305049 may read on the following parts of the GS1 Digital Link Standard Release 1.2:

- 3.4 Structure of the compressed string
- 3.7 Formal ABNF grammar for compressed GS1 Digital Link URIs
- 3.8 Compression procedure and flowcharts
- 3.9 Decompression procedure and flowcharts

A.1.2.2 Patents and Patent Applications from Servicetag SAS

Servicetag SAS of 171 bis, Avenue Charles de Gaulle, 92200 Neuilly-sur-Seine (France), the applicant of the patent application EP 18306189 (for the purpose of this specific paragraph A.1.2.2, the "**Patent Application**"), who participated to the Work Group designing the GS1 Digital Link: Compression Standard Release 1.2, gave notice that they believe their Patent Application contains Essential Claims for the implementation of the GS1 Digital Link: Compression Standard Release 1.2 for the following sections:

- 3 Compression and decompression

If the patent application materialises, Servicetag SAS grants a royalty free license according to the GS1 IP Policy but only as far as methods are described explicitly in the GS1 Digital Link: Compression Standard Release 1.2. Implementations that extend in implementation what is described in GS1 Digital Link: Compression Standard Release 1.2 can contact Servicetag SAS for licensing.

