The Global Language of Business

GS1 Digital Link Standard

Legacy standard serving as an archive and reference point for future related standard developments

Release 1.1.4, Ratified, Dec 2025

Document Summary

Document Item	Current Value
Document Name	GS1 Digital Link Standard
Document Date	Dec 2025
Document Version	1.1
Document Issue	4
Document Status	Ratified
Document Description	Legacy standard serving as an archive and reference point for future related standard developments

Contributors

Name	Organisation
Sprague Ackley	Napeequa Concepts LLC
Makoto Akutagawa	GS1 Japan
Cherise Allison	Sam's Club
David Almroth	GS1 Sweden
Pete Alvarez	GS1 Global Office
Phil Archer (editor)	**GS1 Global Office**
Patrick Arijs	COLRUYT GROUP NV
Karen Arkesteyn	GS1 Belgium & Luxembourg
Joachim Arnold	BASF SE
Koji Asano	GS1 Japan
Paul Ashford	ICCBBA
Pascal Aulagnet	Pfizer
Bo Bäckström	Axfood Sverige AB
Jonas Batt	GS1 Switzerland
Claude Baudry	GS1 France
Jin Bei	GS1 China
Robert Beideman	GS1 Global Office
Dieter Beitz	CSB-System SE
Benjamin Bekritsky	Zebra Technologies Corporation
Zaid Ben Hmad	DECATHLON
Dana Benson	GS1 US
Andree Berg	GS1 Germany
Yann Berthy	Roquette Frères
Mats Bjorkqvist	GS1 Sweden
Dennis Black	BD
Phillip Bocci	Ecolab
Sven Böckelmann	benelog GmbH & Co. KG
Zsolt Bocsi	GS1 Hungary

Name	Organisation
Ardetha Bradley	Georgia Pacific
Laure Breduillieard Brunet	Nestlé
Megan Brewster	Impinj, Inc
Chris Brown	Printronix Auto ID
David Buckley	GS1 Global Office
Heide Buhl	GS1 Germany
Margarita Bykova	GS1 Russia
Jaewook Byun	Auto-ID Labs at KAIST
Elena Campdelacreu	GS1 Spain
Emanuela Casalini	GS1 Italy
Karolin Catela	GS1 Sweden
Tony Ceder	Charmingtrim
Anthony Chan	GS1 Hong Kong, China
Christine Chang	Solventum (previously 3M Healthcare)
Jeffrey Chen	Printronix Auto ID
Raman Chhima	GS1 New Zealand
Francesca Chiovenda	GS1 Global Office
Marjolein Christiaanse	GS1 Netherlands
James Chronowski	GS1 US
Flavia Costa	GS1 Brasil
Benjamin Couty	GS1 France
Amanda Creane	GS1 Ireland
Jeffrey Cree	Ahold (USA)
Jay Crowley	US Data Management, LLC (USDM)
Oscar Cruz	GS1 Mexico
Bence Csordás	GS1 Hungary
Chase Cunningham	Wal-Mart Stores, Inc.
Tarryn Daniels	GS1 South Africa
Kevin Dean	Dolphin Data Development Ltd.
Sean Dennison	GS1 Ireland
Eno Dhamo	GS1 Canada
Peta Ding (editor)	**GS1 Global Office**
Chris Diorio	Impinj, Inc
Xiaowen Dong	GS1 China
Camille Dreyfuss	GS1 France
Han Du	GS1 China
Marcel Ducceschi	GS1 Switzerland
Jeanne Duckett	Avery Dennison RFID
Jt Dutta	DHL
Linden Eagles	GS1 New Zealand
Nordine Eddaoudi	GS1 France

Name	Organisation
Fadi El-Turk	GS1 UK
Alexis Elloso	GS1 Australia
Tomas Eriksson	Byggtrygg AB / XL-BYGG
Juliet Andrea Espinosa Martinez	GS1 Colombia
Filipe Esteves	GS1 Portugal
Dawn Fiorentino Izzi	DoD Logistics AIT Standards Office
Glyn Fogell	Shoprite Checkers
Piotr Frackowiak	GS1 Poland
Lenci François	ALSTOM Ltd
Guido Freijomil	GS1 Argentina
Cameron Frith	GS1 New Zealand
Nigel Fuller	GS1 Australia
Andreas Füßler	GS1 Germany
Stefan Gathmann	GS1 Ireland
Haifa Gaza	Auto-ID Labs at KAIST
Alexander Gerasimenko	Mars, Inc.
Jean-Christophe Gilbert (co-chair)	**GS1 France**
Vanessa Giulieri	GS1 Italy
Oumy Gning	GS1 Global Office
Chiang Fein Goh	GS1 Malaysia
Nicole Golestani	GS1 Canada
Carl Gomberg	Premier Inc
Alan Gormley	GS1 Ireland
Heinz Graf	GS1 Switzerland
Nadi (Scott) Gray	GS1 Global Office
Christer Green	SBJF Service AB
Jonathan Gregory	GS1 US
Gerald Gruber	GS1 Austria
Dominique Guinard	Digimarc
Kai Hachmann	Edeka Zentrale AG & Co. KG
David Hackbarth	Procter & Gamble Co.
Michaela Hähn	GS1 Germany
Elzbieta Halas	GS1 Poland
Dominik Halbeisen	GS1 Switzerland
Eileen Harpell	GS1 Global Office
Mark Harrison (editor)	**Milecastle Media Limited**
Gary Hartley	GS1 New Zealand
Dave Harty	Adents
Andrew Hearn	GS1 Global Office
Natasha Helleur	GS1 New Zealand
Olle Hellman	GS1 Sweden

Name	Organisation
Lindsay Hillman	GS1 US
Rob Hoffman	Hershey Company (The)
Bernie Hogan	Independent Consultant - Bernie Hogan
Frank Hogema	GS1 Netherlands
Sandra Hohenecker	GS1 Germany
Lars Kristen Holst	Moelven Wood AB
Christine Horvath-Hanko	GS1 Hungary
Agata Horzela	GS1 Poland
Hitoshi Hoshino	GS1 Japan
Tomi Ihalainen	GS1 Finland
Marc Inderbitzin	Migros-Genossenschafts-Bund
José Ramón Islas	GS1 Mexico
Gunnar Ivansson	Learningwell AB
Yoshihiko Iwasaki	GS1 Japan
Guilda Javaheri	Golden State Foods
Ed Jesus	Chep
Han Jie	GS1 China
Peter Johnson	Procter & Gamble Co.
Peter Jönsson	GS1 Sweden
Deborah Joplin	GS1 New Zealand
Nora Kaci	GS1 Global Office
Martin Kairu	GS1 South Africa
Hitomi Kajita	GS1 Japan
Nicole Kaller	DuPont Healthcare
Iliada Karali	GS1 Association Greece
Katalin Kecskés	GS1 Hungary
Steven Keddie	GS1 Global Office
Audrey Kelly	GS1 Global Office
John Keogh	Shantalla Inc
Kimmo Keravuori	GS1 Finland
Mads Kibsgaard	GS1 Denmark
Edmund Kienast	Australian Digital Health Agency
Kazuna Kimura	GS1 Japan
Sabine Klaeser	GS1 Germany
Kasia Koblak-Smith	DHL
Phyllis Koch	The Schwan Food Company
Catherine Koetz	Australian Digital Health Agency
Bojan Kovačič	GS1 Slovenia
Zoltan Krazli	GS1 Hungary
Alexey Krotkov	GS1 Russia
Tomas Kubicek	smart-tec GmbH & co. KG

Name	Organisation
Freddy Ladino Perdomo	GS1 Colombia
Huyen Le	GS1 Vietnam
Lyndon Lee	Tesco Stores Ltd.
Kevin Lee	GS1 Korea
Zhimin Li	GS1 China
Ildikó Lieber	GS1 Hungary
Ildikó Lieber Coe	GS1 in Europe
Tina Lin	GS1 Chinese Taipei
Nicolas Liou	GS1 Chinese Taipei
Johan Lisemark	Vida Wood AB
Yixuan Liu	GS1 China
Wei Liu	GS1 China
Xiaoyan Liu	GS1 China
Huiru Lou	GS1 China
Marisa Lu	GS1 Chinese Taipei
Wayne Luk	GS1 Hong Kong, China
Yan Luo	GS1 China
Chatchanok Lupakchee	GS1 Thailand
Ning Ma	GS1 China
Ilka Machemer	GS1 Germany
Fumi Maekawa	GS1 Japan
Patrick Main	Cook Medical Inc.
Ralph Maresco	Procter & Gamble Co.
Timothy Marsh	MCL
Ryan Mavin	ACT Health
Mike Meakin	DHL
Ned Mears	GS1 US
Terje Menkerud	GS1 Norway
Edward C Merrill	GS1 Global Office
Joanne Metcalf	Essity North America Inc
Andrew Meyer	GS1 US
Holly Mitchell	Seagull Scientific
Federico Mittersteiner	GS1 Italy
Ephraim Mokheseng	GS1 South Africa
Karen Moniz	ICCBBA
Doug Moody	PepsiCo, Inc.
Julie Moore	Mead Westvaco
Andrew Morehead	GS1 US
Gena Morgan	GS1 US
Reiko Moritani	GS1 Japan
Robert Moss	FIP HPS

Name	Organisation
Nolwandle Mthiyane	GS1 South Africa
Alice Mukaru	GS1 Sweden
Dan Mullen	GS1 Global Office
Paul Muller	EM Microelectronic
Mori Naoko	GS1 Japan
Nico Nardozi	AbbVie S.r.l.
Safae Nassih	GS1 France
Zubair Nazir	GS1 Canada
Giada Necci	GS1 Italy
Steven Nelson	Tyson
Denis O'Brien	GS1 Ireland
Marcelo Oliveira Sa	GS1 Brasil
Hirra Oppal	GS1 UK
Michel Ottiker (co-chair)	**GS1 Switzerland**
Luisa Ovalle	GS1 Colombia
Dimi Pachiyannis	GS1 Australia
Michele Padayachee	GS1 South Africa
Alexander Pakhomov	LLC "Center for the Development of Advanced Technologies"
Chensheng Pan	GS1 China
Kyle Parker	Mitas Corporation
Sergio Pastrana	GS1 Mexico
Leonel Pava	GS1 Colombia
John Pearce	Axicon
James Perng	GS1 Chinese Taipei
Sarina Pielaat	GS1 Netherlands
Manfred Piller	GS1 Austria
Neil Piper	GS1 Global Office
Zhangli Pollet	GS1 Belgium & Luxembourg
Josef Preishuber-Pflügl	innobir e.U.
Albertus Pretorius	Tonnjes ISI Patent Holding GmbH
Mathias Prost	GS1 France
Matheus Quadros	GS1 Brasil
Aruna Ravikumar	GS1 Australia
Hajo Reissmann	Hajo Reissmann Consulting
Eduardo Alberto Remigio Munguia	GS1 Mexico
Craig Alan Repec	GS1 Global Office
Nicolas Resier	GS1 Belgium & Luxembourg
William Reynolds	Camcode Global
Diane Riccardi	Johnson & Johnson
Pharapa Romphothong	GS1 Thailand
Pere Rosell Plajats	GS1 Spain

Name	Organisation
Don Roskowiak	Target Corporation
Greg Rowe	GS1 Global Office
Uwe Ruedel	GS1 Switzerland
Michiel Ruighaver	GS1 Australia
Liu Ruizhi	GS1 China
Zbigniew Rusinek	GS1 Poland
Bonnie Ryan	GS1 Australia
John Ryu (facilitator)	**GS1 Global Office**
Zbigniew Sagan	Advanced Track and Trace
Hemant Sahgal	IRIS Software
Sofia Salcedo	Logyca
Alexander Sanchez	GS1 Mexico
George Sarantavgas	GS1 Association Greece
Mayu Sasase	GS1 Japan
Yuki Sato	GS1 Japan
Sue Schmid	GS1 Global Office
Lori Schrop	GS1 Global Office
Eugen Sehorz	GS1 Austria
Xinyu Shi	GS1 China
Yuko Shimizu	GS1 Japan
Marcel Sieira	GS1 Australia
Fernando Silveira	GS1 Brasil
Palma Simbari	Bayer AG - Division Pharma
Steven Simske	Colorado State University
Chumisa Sizathu	GS1 South Africa
Jens Slama	GS1 Germany
Brandi Smith	GS1 US
Tania Snioch	GS1 Global Office
Olga Soboleva	GS1 Russia
Jim Springer	EM Microelectronic
Roko Staničić	GS1 Slovenia
Andrew Steele	GS1 Australia
Sylvia Stein	GS1 Netherlands
Christa Suc	GS1 UK
Georgette Suggs	Sitation, LLC.
Diane Taillard	GS1 Global Office
Hiromitsu Takai	GS1 Japan
Taishi Takaoka	GS1 Japan
Claude Tetelin	GS1 Global Office
Sonya Thomas	Sam's Club
Stefan Thomas	GS1 Germany

Name	Organisation
Gina Tomassi	PepsiCo, Inc.
Laurent Tonnelier	mobiLead
Viet Tran	GS1 Vietnam
Ed Treacy	Produce Marketing Association (PMA)
Ralph Troeger	GS1 Germany
Jangchup Tsechung	Migros-Genossenschafts-Bund
Alec Tubridy	GS1 Australia
Jesse Tuominen	Voyantic Ltd
Toshihide Ueda	GS1 Japan
Frits Van Den Bos	GS1 Netherlands
Ray Vaughan	Omron Microscan Systems Inc.
Rodrigo Vaz	CPA Wernher von Braun
Cindy Velázquez	GS1 Mexico
Audun Vennesland	SINTEF
Andrew Verb	Bar Code Graphics, Inc.
Charles-Francois Vermeesch	SNCF Rolling Stock Department
Ricardo Verza Amaral Melo	GS1 Brasil
Jaco Voorspuij	CONAS BV
Vishnu Vyas	DuPont Healthcare
Hanna Walczak	GS1 Poland
Amber Walls	Crimson Raft Consulting
Lei Wang	GS1 China
Chunguang Wang	GS1 China
Lynn Wang	GS1 China
Dayou Wang	Zebra Technologies Corporation
Wenyu Wang	GS1 China
Jiayi Wang	GS1 China
Di Wang	GS1 China
Metinee Wanlayangkoon	GS1 Thailand
Ethan Ward	GS1 Australia
John Weatherby	JDHW Consulting
Wilfried Weigelt	REA Elektronik GmbH
Kathy Welch	Wegmans Food Markets
Tasha Wiehe	GS1 Global Office
Kumoro Wijanarko	GS1 Indonesia
Ola Wilhelmsson	Norra Timber
Roman Winter	GS1 Germany
Melanie Wishart	GS1 Australia
Agata Witkowska	GS1 Poland
George Wright Iv	Product Identification & Processing Systems
Xinmin Wu	GS1 China

Name	Organisation
Juan Wu	GS1 China
Huang Xin	GS1 China
Ruoyun Yan	GS1 China
Xie Yi	GS1 China
Charles Young	Solidsoft Reply
Shi Yu	Beijing REN JU ZHI HUI Technology Co. Ltd.
Ying Yu	GS1 China
Victor Zhang	GS1 China
Yuan Zhang	GS1 China
Li Zheng	GS1 China
Bo Zhou	GS1 China
Eric Zhuo	GS1 Singapore
Sergio Zúñiga	GS1 Mexico
亭杰 申	GS1 China

Log of Changes

Release	Date of Change	Changed By	Summary of Change
1.0	Aug 2018	Phil Archer, Dominique Guinard, Mark Harrison, Marie Petre & Greg Rowe	Initial release developed on WR 17-000343. Originally published under the title *GS1 Web URI Structure Standard*
0.1	2019-01-29	Phil Archer	Preparation of original material for addition of new sections
0.2	2019-03-05	Phil Archer & David Buckley	Integration of resolver chapter, term 'GS1 Web URI' replaced with GS1 Digital Link URI as appropriate throughout. Minor changes not tracked. New sections highlighted in blue. Specific issues raised throughout the document. Updated to GS1 Branding.
0.3	2019-04-17	Dominique Guinard	Added sections about CORS and TLS.
0.4-0.18	2019-04-18 – 2019-06-05	Mark Harrison, Phil Archer, Laurent Tonnelier, Dominique Guinard, Grant Courtney, Tony Rodriguez et al	Addition of chapter on compression/decompression, Addition of semantics chapter Evolutionary steps as decided by the working group concerning, for example, link type being the default, cf. 'a link', introduction of concept pf a class 2 resolver, allowance for resolvers to cover subset of GS1 identifier space, definition of the resolver description file, inclusion of a conformance statement. Addition of AIs 417, 235 and 7040
0.19	2019-06-10	Marie Petre, Rigo Wenning	Addition of disclaimer ahead of public Community Review

Release	Date of Change	Changed By	Summary of Change
0.20	2019-08-15	Mark Harrison, Phil Archer, Dominique Guinard, Laurent Tonnelier, Grant Courtney addressing additional comments from: Ralph Tröger, Ilka Machemer, John Pearce, Bruno Lai, Roland Donzelle, Mark Nottingham, Hajo Reissmann, Terry Burton, Steven Keddie, Dominique Guinard, Menno Bruil, Sandun Jayawardena, Frits van den Bos	Changes following Community Review
0.21	2019-10-10	Phil Archer	Further post CommRev changes
0.22	2019-12-10	Phil Archer	Post IP Review changes
1.1	Feb 2020	Phil Archer, Mark Harrison, Greg Rowe – plus all above	Updates based upon WR 18-231 which can be found in section 4
1.1.1	Jun 2021		WR 22-176 errata
1.1.2	Nov 2022	Phil Archer, Mark Harrison	WR 22-336 and WR 22-337 errata
1.1.3	Mar 2024	Phil Archer	WR 24-055 adding note about conformant resolver standard
1.1.4	Oct 2025	Phil Archer, Mark Harrison, Peta Ding	WR 25-265 Removal of sections superseded by other standards. Clarifying nature of remaining sections. Updates to Table F and Table P to reflect all current GS1 Application Identifiers appearing in GS1 General Specifications v25 and subsequent ratified GSCNs. Updates to Table Opt to designate previously reserved compression headers '7A' and '7B' for use in expressing EPC binary strings within a special compressed GS1 Digital Link URI syntax – see "GS1 Digital Link URI: Compression Technical Standard for EPC binary strings" for further details. WR 25-448 Style guide alignment addressed as errata fixes

Disclaimer

GS1 seeks to minimise barriers to the adoption of its standards and guidelines by making the intellectual property required to implement them available, to the greatest extent possible, on a royalty-free basis, or when necessary, under a RAND licence. Such royalty-free and RAND licences are provided pursuant to the GS1 IP Policy (available here: https://www.gs1.org/standards/ip), which governs the work of work group participants who contribute to drafting standards and guidelines, including this document. In addition to licences, the GS1 IP Policy provides various benefits and obligations that apply to all implementers of GS1 standards and guidelines, and all implementations of GS1 standards are subject to those terms.

Nevertheless, please note the possibility that an implementation of one or more features of this standard or guideline may be the subject of a patent or other intellectual property right that is not covered by the licences granted pursuant to the IP Policy. In addition, the licences granted under the IP Policy do not include the IP rights or claims of third parties who were not participants in the corresponding standard development work group.

Accordingly, GS1 recommends that any person or organisation developing an implementation of this standard or guideline should determine whether any patents or other intellectual property may encompass such implementation, and whether a licence under a patent or other IP right is needed. The implementer should determine the potential need for licensing in view of the details of the specific implementation being designed in consultation with that party's patent counsel.

The official versions of all GS1 standards and guidelines are provided as PDF files on GS1's online reference directory (https://ref.gs1.org) (the "GS1 Reference"). Any other representations of standards or guidelines in any other format (e.g., web pages) are provided for convenience and descriptive purposes only, and in the event of a conflict, the GS1 Reference document shall govern.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR PARTICULAR PURPOSE, ACCURACY OR COMPLETENESS, OR ANY WARRANTY OTHERWISE ARISING OUT OF THIS DOCUMENT. GS1 disclaims all liability for any damages arising from any 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 makes no commitment to update the information contained herein, and retains the right to make changes to this document at any time, without notice. GS1® and the GS1 logo are registered trademarks of GS1 AISBL.

Table of Contents

1 Introduction

This section is informative

Beginning in 2014, the original work to create what became "GS1 Digital Link" was 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.

Today, these ideas are realised in the much broader Global Migration to 2D programme. In that context, it has been helpful to separate out the various components of that original work and to develop them independently of each other, taking account of evolving business needs and implementation experience.

Early versions of "the GS1 Digital Link standard" in August 2018 and February 2020 included several chapters that have since become discrete standards. Those standards are:

- GS1 Digital Link Standard: URI Syntax [DL-URI]. This foundational standard includes formal definitions, explanations and examples of how to encode GS1 identifiers in a Web URI structure. The term "GS1 Digital Link" should now be seen as synonymous with the GS1 Digital Link Standard: URI Syntax.
- GS1-Conformant Resolver Standard [Resolver]. A GS1 Digital Link URI may connect to an online service that can *resolve* the encoded GS1 identifiers and redirect queries to any number of information resources that are related to the identified entity. Links are described using link relation types and other metadata.

The chapters in the original GS1 Digital Link standard that have been superseded by those discrete standards have been removed from this document entirely. The remaining chapters cover details not currently defined elsewhere. These are:

- Details of a compression and decompression algorithm that, in some circumstances, can reduce the number of characters in a GS1 Digital Link URI. This technical chapter is not referenced in any GS1 Application Standard and there is no expectation that the compressed form of a GS1 Digital Link URI as defined by this standard will ever be used in open supply chains. This is a legacy specification that is not expected to be maintained after version 1.1.4. However, it is necessary as a background reference for the GS1 Digital Link URI: Compression Technical Standard for EPC binary strings [DL-C].
- Definitions of the semantics of some of the GS1 Application Identifiers, expressed as terms in the GS1 Web Vocabulary. This work is expected to be completed as part of a broader effort to define the semantics of all GS1 Application Identifiers. On completion of such work, the semantics chapter will be removed from future versions of this document.

1.1 Convenience alphas used in this document

The remaining normative chapters of this document are unchanged since the February 2020 publication of the "all-in-one" GS1 Digital Link Standard [DL-11]. This means that **some of the text is out of date**. In particular, there are multiple references to "convenience alphas" such as `gtin` and `gln` that, in the original "all-in-one" GS1 Digital Link Standard, could be used instead of the equivalent GS1 Application Identifiers. This feature was deprecated in version 1.2.0 (January 2021) of the GS1 Digital Link Standard: URI syntax and removed entirely in version 1.3.0 (November 2022).

It follows that new implementations of any part of this standard use convenience alphas.

2 Compression and decompression

This section and all subsections are normative except where indicated

2.1 Important note about the status of this chapter

This section is informative

The compression/decompression algorithm defined in this chapter is a Technical Standard (how), as distinct from an Application standard (where used). The algorithm and the tables defined here are legacy specifications, that will no longer be maintained after version 1.1.4. However, they ensure non-collision with the GS1 Digital Link URI: Compression Technical Standard for EPC binary strings [DL-C] which defines the use of the compression header values '7A' and '7B' that were reserved in this standard. These result in compression strings that begin with 'eh' or 'ex', indicating an EPC binary string encoded in lower case hexadecimal or file-safe URI-safe base 64 characters, respectively, which can then be decoded to an uncompressed GS1 Digital Link URI using the GS1 EPC Tag Data Standard [TDS] or implementations of the GS1 EPC Tag Data Translation Standard [TDT].

Compressed GS1 Digital Link URIs that use the algorithm defined in this chapter, are not expected to be used in any future Application Standard and are not approved for use in open supply chain standards. Only compressed GS1 Digital Link URIs using the 'eh' or 'ex' flags defined by the GS1 Digital Link URI: Compression Technical Standard for EPC binary strings [DL-C] are expected to be used in any future Application Standard and approved for use in open supply chain standards (see note below). Please refer to the GS1 General Specifications for the relevant Application Standards and the GS1 Digital Link URI: Compression Technical Standard for EPC binary strings [DL-C] for further information on compressed GS1 Digital Link URIs.

Note: To minimise the disruption of encountering unexpected / unsupported AIDC data carriers or AIDC data carrier syntaxes in open supply chains, GS1 performs Policy B11 assessments. Once a B11 assessment is approved, use of the new AIDC data carrier or data carrier syntax within an Application Standard (as an exclusive or additional data carrier) is subject to the Global Standards Management Process (GSMP).

2.2 Purpose of compression

This section is informative

GS1 Digital Link URIs can be used natively with any data carrier that can directly embed an entire URL. These include NFC tags, regular ISO 18004 QR codes, digital watermarks, Data Matrix codes, etc. It is not possible to natively encode GS1 Digital Link URIs verbatim within EPC/RFID tags without causing potential collisions with existing tags encoded with binary encodings of EPC identifiers. The recently ratified GS1 Digital Link URI: Compression Technical Standard for EPC binary strings [DL-C] enable a compressed form of the GS1 Digital Link URI to express an EPC binary string that has been read from an EPC/RFID tag. Please refer to GS1 Digital Link URI: Compression Technical Standard for EPC binary strings [DL-C] for further information.

Compression as defined in this section results in a compressed GS1 Digital Link URI that is shorter in length. For optical data carriers, this can 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.

Note that the compression algorithm focuses on reducing the number of characters in the URI without taking account of how those characters are encoded in optical data carriers. Digits are more efficiently encoded in data carriers than uppercase alphas and some punctuation characters which in turn are more efficiently encoded than lower case alphas and other characters. Taking into account these details it is possible to create a compressed GS1 Digital Link URI that is shorter than the original, however that does not necessarily reduce the final size of the GS1 compliant 2D barcode.

If the aim of using the compression/decompression algorithm defined here is to minimise the size of the resultant QR Code or Data Matrix, the GS1 Retail 2D barcode size estimator [EST] is

recommended as a tool to determine whether this will have the desired effect. It can be used, for example, to show that using only uppercase letters and no lowercase letters wherever possible is often a more effective way to reduce barcode size than simply reducing the character count.

2.3 Technical requirements extracted from the business requirements

This section is informative

The Business Requirements Analysis Document (BRAD) 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.

2.4 Viability of types of compression techniques and opportunities for compression

This section is informative

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.

2.5 Structure of the compressed string

2.5.1 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.

2.5.2 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				
03	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			
235	X		28			
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	N	6				

3290-3295						
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				
3950-3955	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 - 4301	X		35			
4302 - 4306	X		70			
4307	X	2				
4308	X		30			
4309	N	20				
4310 - 4311	X		35			
4312 - 4316	X		70			
4317	X	2				
4318	X		20			
4319	X		30			
4320	X		35			
4321 – 4323	N	1				
4324 – 4325	N	10				
4326	N	6				
4330 – 4333	N	6		X		1
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			
7011	N	6		N		4
7020	X		20			
7021	X		20			
7022	X		20			
7023	X		30			
7030-7039	N	3		X		27
7040	N	1		X	3	
7041	X		4			
710 - 717	X		20			
7230-7239	X		30			
7240	X		20			
7241	N	2				
7242	X		25			
7250	N	8				
7251	N	12				
7252	N	1				
7253	X		40			
7254	X		40			
7255	X		10			
7256	X		90			
7257	X		70			
7258	X	3				
7259	X		40			
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			
8014	X		25			
8017	N	18				
8018	N	18				
8019	N		10			
8020	X		25			
8026	N	18				
8030	X		90			
8040	N	15				
8041	N	15				
8042	N	32				
8043	N	18		N		2
8110	X		70			
8111	N	4				
8112	X		70			
8200	X		70			
90	X		30			
91-99	X		90			

2.5.3 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 2.6) 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	**17**	2	**33**	4	**70**	4	**94**	2
01	2	**20**	2	**34**	4	**71**	3	**95**	2
02	2	**21**	2	**35**	4	**72**	4	**96**	2
03	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		

2.5.4 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

2.5.4.1 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.**

2.5.4.2 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.

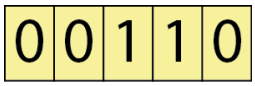
A length indicator is used where the length of a value or value component is not of predefined length.

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 (L_{max}) for the value or value component. A length indicator of n bits permits expression of a length in the range 0 to ($2^n - 1$).

For convenience, the following table provides value for n for 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_v=	n_v = 4n	n_v = 6n	n_v = 7n

n	ceiling(n * log(10)/log(2))			

2.6 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.

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
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*	
7A	EPC binary string – option A (expressed as hexadecimal encoding) See "GS1 Compression technical standard for EPC binary strings" for details		
7B	EPC binary string – option B (expressed as file-safe base 64 encoding) See "GS1 Compression technical standard for EPC binary strings" for details		
7C – 7F	*reserved*	*4 values reserved for future use*	
80 – 8F	*reserved*	*16 values reserved for future use*	
9E , 9F	*reserved*	*2 values reserved for future use*	
AA – AF	*reserved*	*6 values reserved for future use*	
B0 – BF	*reserved*	*16 values reserved for future use*	
CE , CF	*reserved*	*2 values 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	

2.7 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.

2.7.1 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.

2.7.1.1 Binary encoding example for GTIN: (01)10614141123459

GS1 AI (01)								Value of AI (01) is fixed-length numeric and left-padded to a total of 47 bits = ceiling (14*log(10)/log(2))																																												
0				1				10614141123459																																										
0	0	0	0	0	0	0	1	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

2.7.1.2 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

2.7.2 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.

2.7.2.1 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

2.7.3 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 2.5.4.1. 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.

2.7.3.1 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

2.7.3.2 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.

<table>
<tr><td colspan="8">GS1 AI (10)</td><td colspan="3">Encoding</td><td colspan="5">Actual Length</td><td colspan="48">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</td></tr>
<tr><td colspan="4">1</td><td colspan="4">0</td><td colspan="3">3</td><td colspan="5">8</td><td colspan="6">X</td><td colspan="6">Y</td><td colspan="6">Z</td><td colspan="6">9</td><td colspan="6">8</td><td colspan="6">7</td><td colspan="6">6</td><td colspan="6">5</td></tr>
<tr><td>0</td><td>0</td><td>0</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>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</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>1</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>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
</table>

2.7.3.3 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.

<table>
<tr><td colspan="8">GS1 AI (10)</td><td colspan="3">Encoding</td><td colspan="5">Actual Length</td><td colspan="32">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</td></tr>
<tr><td colspan="4">1</td><td colspan="4">0</td><td colspan="3">2</td><td colspan="5">8</td><td colspan="4">A</td><td colspan="4">B</td><td colspan="4">C</td><td colspan="4">9</td><td colspan="4">8</td><td colspan="4">7</td><td colspan="4">6</td><td colspan="4">5</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</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>1</td><td>0</td><td>0</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>1</td><td>1</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>0</td><td>0</td><td>0</td><td>1</td><td>1</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></tr>
</table>

2.7.3.4 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.

<table>
<tr><td colspan="8">GS1 AI (10)</td><td colspan="3">Encoding</td><td colspan="5">Actual Length</td><td colspan="27">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</td></tr>
<tr><td colspan="4">1</td><td colspan="4">0</td><td colspan="3">0</td><td colspan="5">8</td><td colspan="27">12398765</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><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>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</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>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
</table>

2.7.4 Optimisation using pre-defined sequences of GS1 Application Identifiers

2.7.4.1 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: 0001110001011011110010110000111100

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	0

In this example, 16 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).

2.8 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 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 2.5.1.

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


compressedSegment = 1*uriSafeBase64char
```

Section 2.8.1 defines the formal ABNF grammar for partially compressed GS1 Digital Link URIs. Section 2.8.2 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 2.10 and flowcharts D1-D4.

2.8.1 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/gtin/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 ‘gtin’ (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 section 2.5.1.

```
primaryIDcomponent = gtin-comp / itip-comp / gmn-comp / cpid-comp
                     / shipTo-comp / billTo-comp / purchasedFrom-comp
                     / shipFor-comp / gln-comp / payTo-comp / glnProd-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
```

2.8.2 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
```

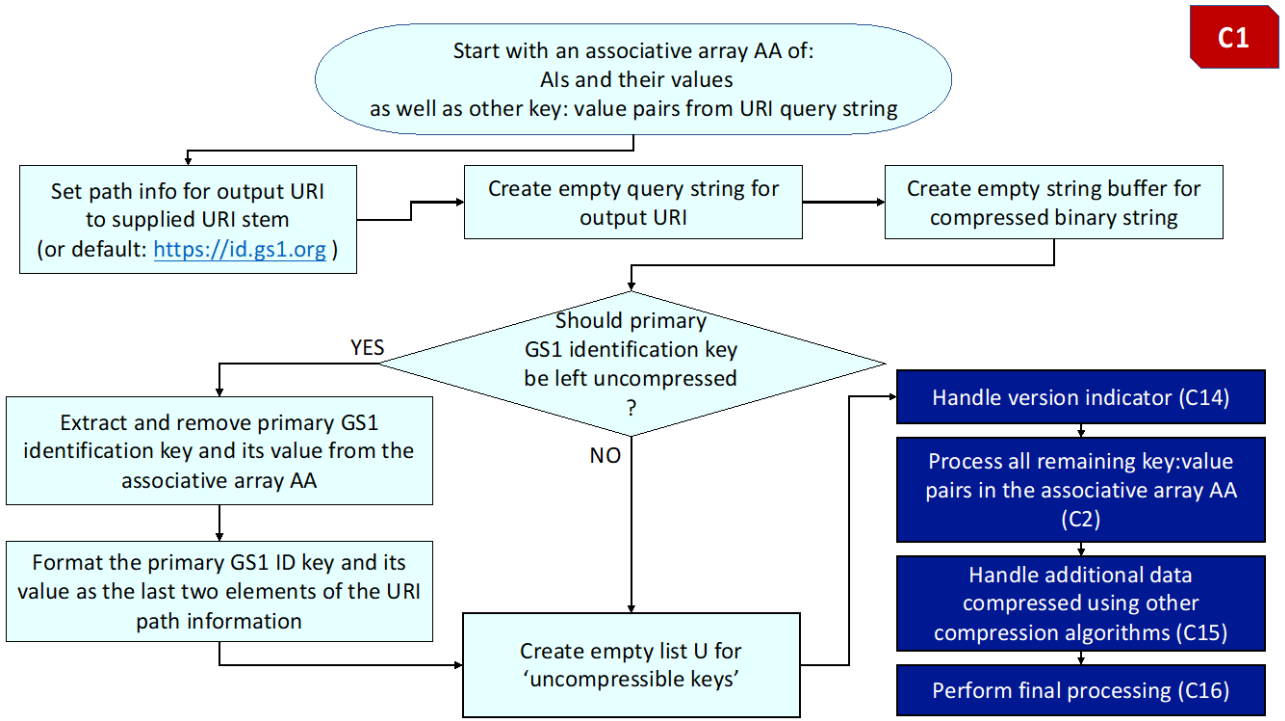
In the example, the final component of the URI path information is the compressed segment ‘DgnYUc1gmji3NU0IREGFDTK2LJm’ consisting of characters only from the URI-safe base 64 alphabet.

```
fullyCompressedGS1webURIPattern
           = compressedSegment [queryStringComp]


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


fullyCompressedCustomGS1webURI
     = customURIstem  fullyCompressedGS1webURIPattern
```

2.9 Compression procedure and flowcharts

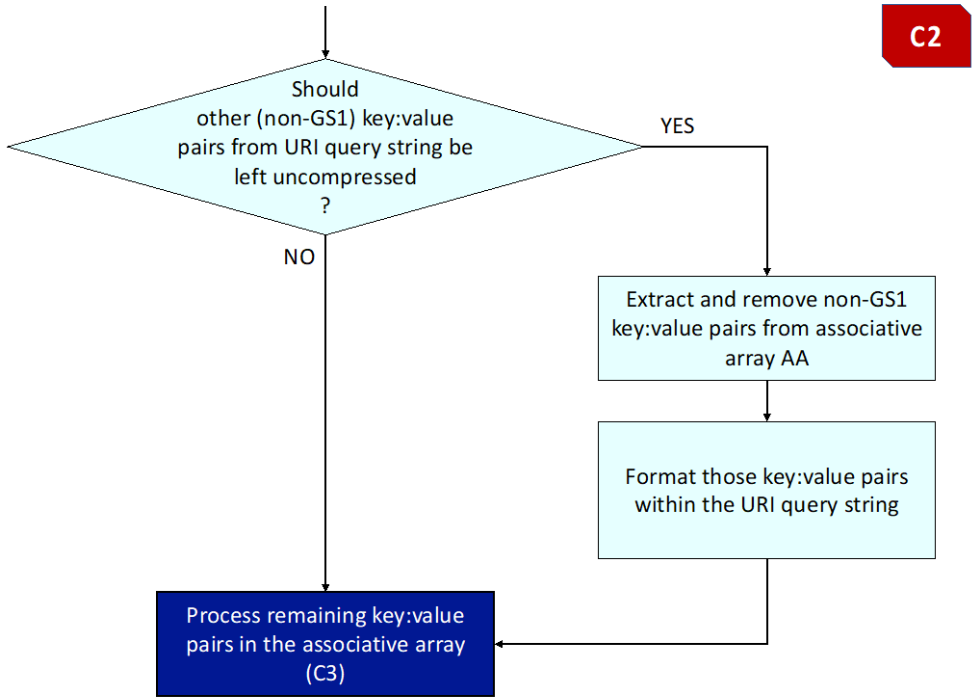
This section provides a set of flowcharts to describe the compression procedure to obtain fully or partially compressed GS1 Digital Link URIs.

Figure C1: Top-level flowchart for v1.0 standard compression of GS1 Digital Link URIs



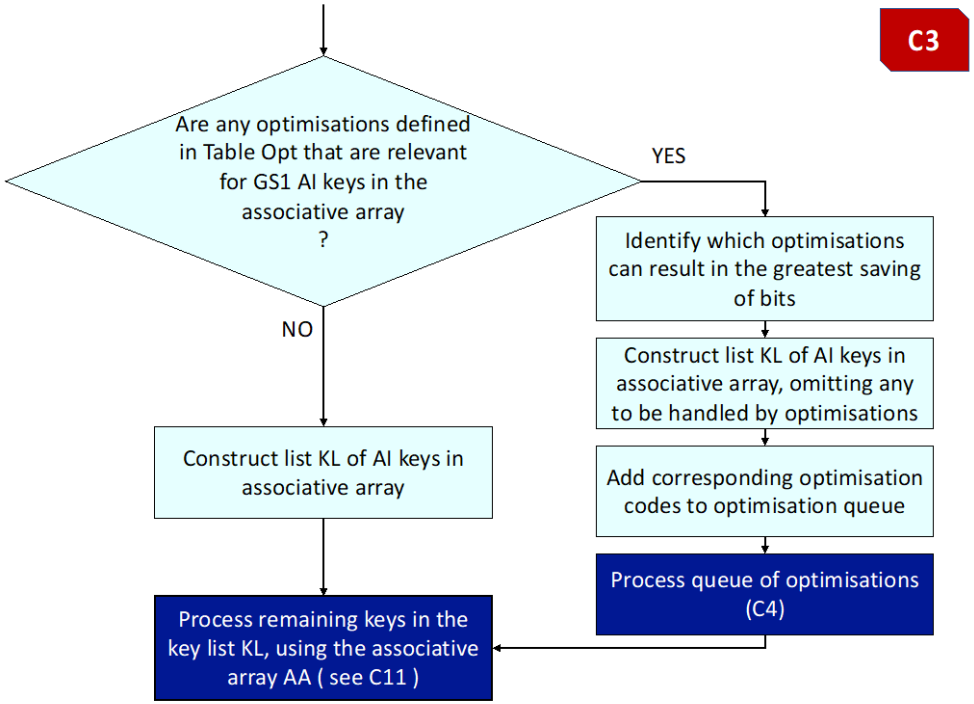
Flowchart C1 provides the high-level flowchart for compression of GS1 Digital Link URIs, starting with an associative array of key:value pairs from element strings or other URI query string key:value pairs. The first decision is whether the primary identifier and its value should be left uncompressed (as is the case for a partially compressed GS1 Digital Link URI). Flowcharts C14, C2, C15 and C16 and their dependents are referenced for further details.

Figure C2: Support non-GS1 key:value pairs from URI query string being compressed or left uncompressed in the URI query string



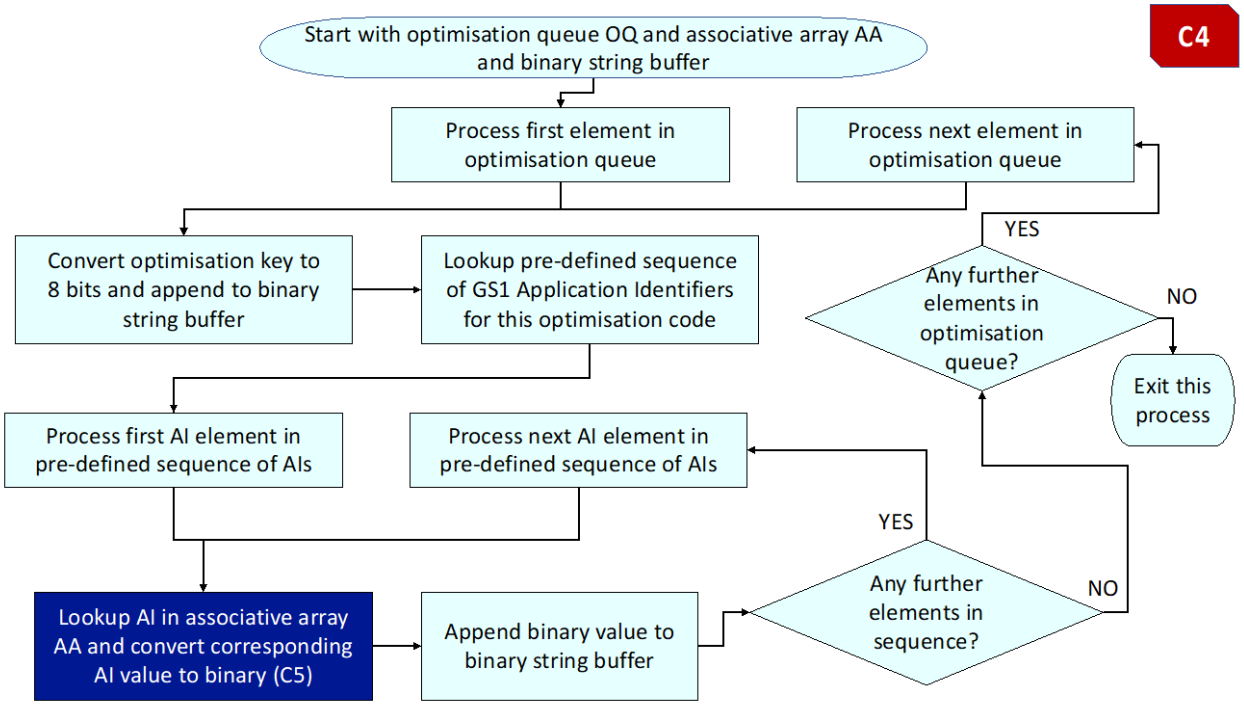
Flowchart C2 explains that if non-GS1 key:value pairs from the URI query string should be compressed, then these must be extracted from the URI query string and included within the associative array. Further processing then moves to flowchart C3.

Figure C3: Support for optimisations using pre-defined sequences of GS1 Application Identifiers – refers to flowchart C4 for further detail



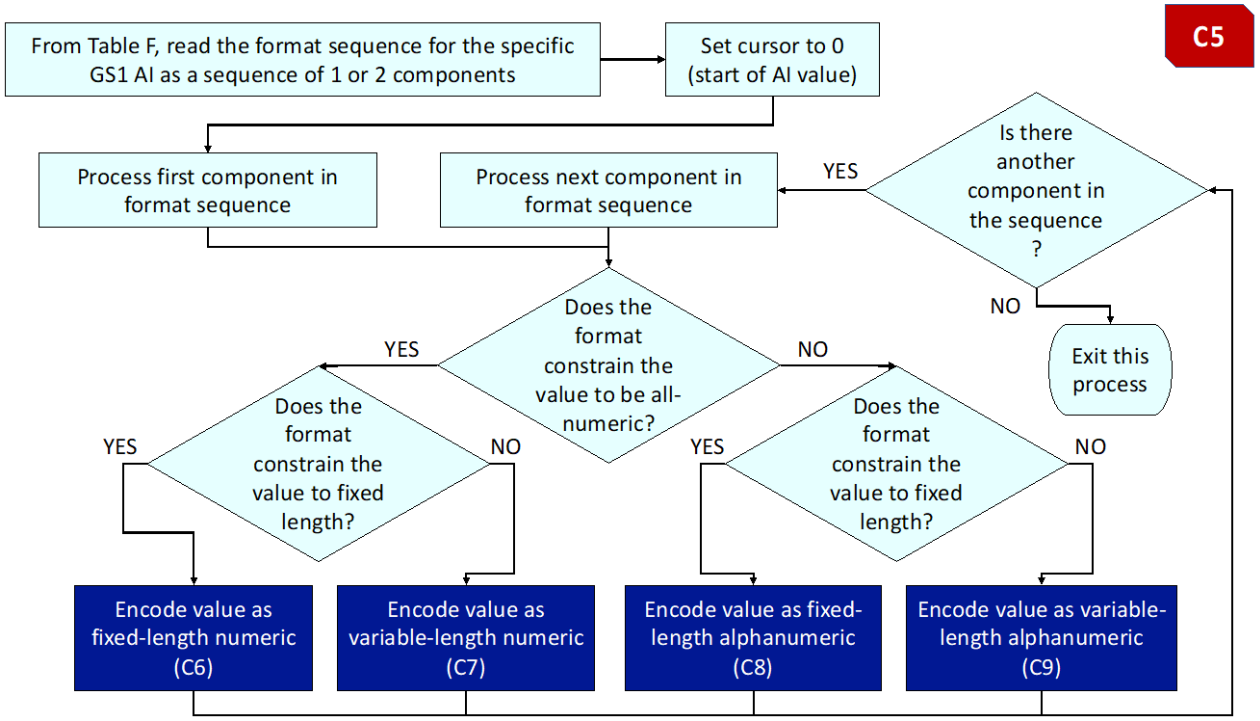
In Flowchart C3, the keys within the associative array are compared against available optimised pre-defined sequences of GS1 AIs, defined in Table Opt. If such optimisations are found, the combination of optimisations that results in the largest total saving of bits is selected and an updated list of AI keys is prepared, omitting those which will be handled via optimisation sequences. Flowchart C4 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 C11.

Figure C4: Process queue of optimisations for pre-defined sequences of GS1 Application Identifiers



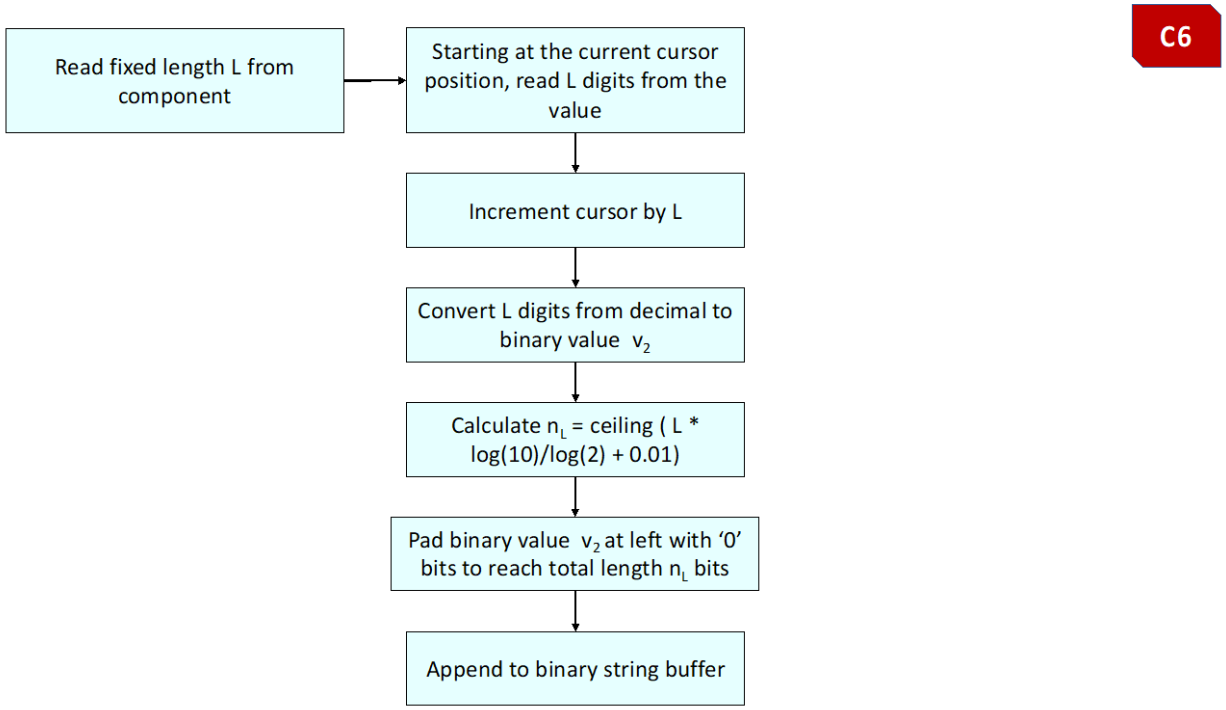
Flowchart C4 explains processing of a queue of optimisations for pre-defined sequences of GS1 Application Identifiers. For each optimisation, the GS1 Application Identifiers are not encoded in binary using 8, 12 or 16 bits per GS1 AI key. Instead a single 8-bit key is used to represent the entire pre-defined sequence. Flowchart C4 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 C4 references Flowchart C5 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 C5: Convert the value of a GS1 Application Identifier to binary



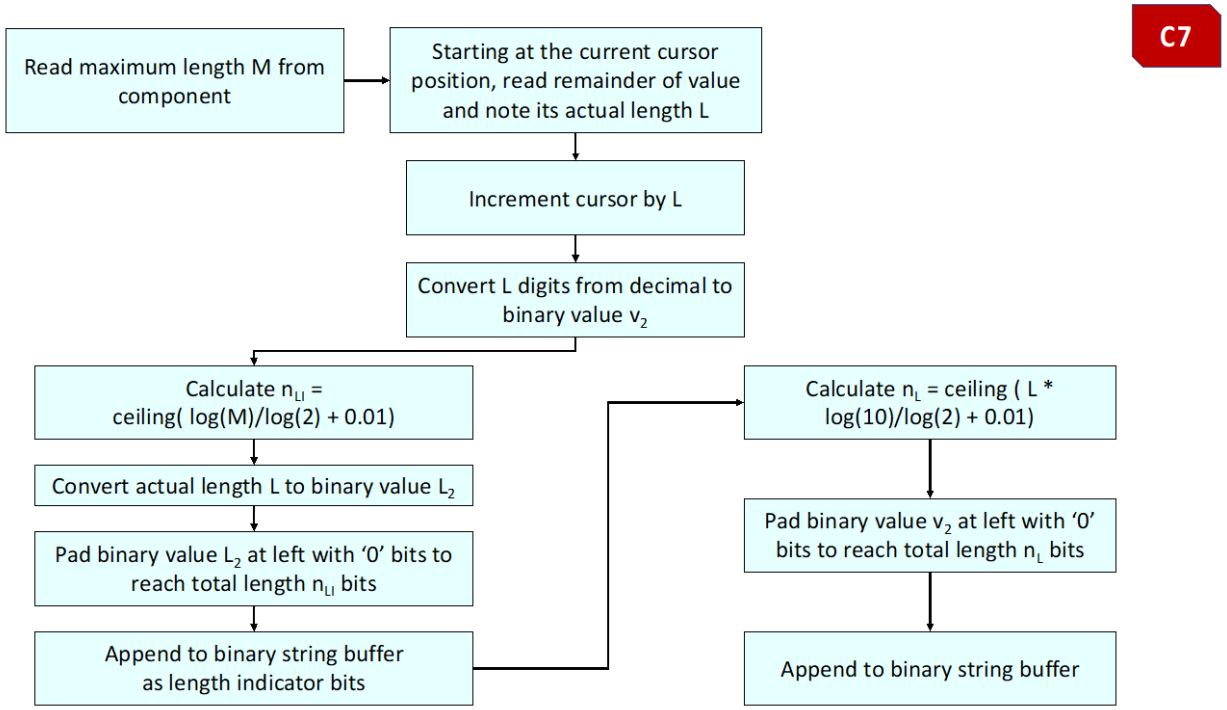
Flowchart C5 explains how the value of each GS1 Application Identifier is encoded in binary. The first step is to find an entry in Table F (the format table) for the GS1 AI key (e.g. 01), which is typically expressed as one or two components, which are processed in turn. For each of these two components, 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 (C6), variable-length numeric (C7), fixed-length alphanumeric (C8) or variable-length alphanumeric (C9). 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 C6: Encode a fixed-length numeric value



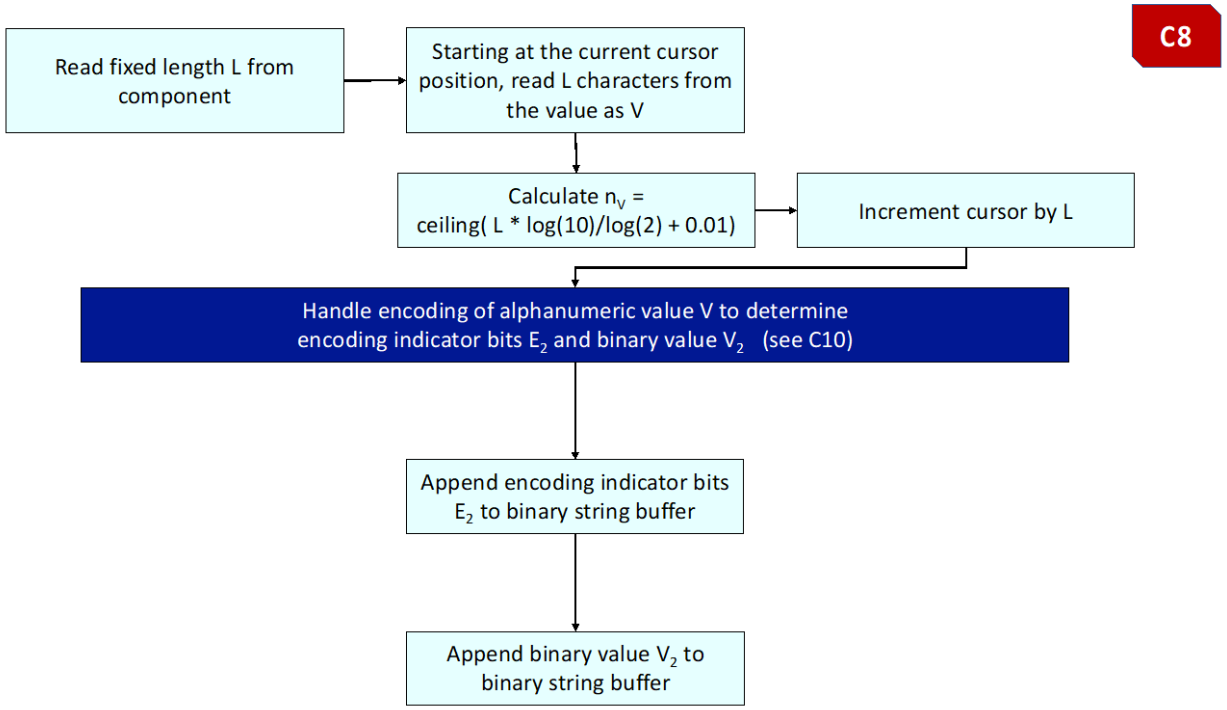
Flowchart C6 explains how to encode a fixed-length numeric value. The number of bits expected is given by the formula for n_L and the binary value is left-padded with '0' bits to reach length n_L.

Figure C7: Encode a variable-length numeric value



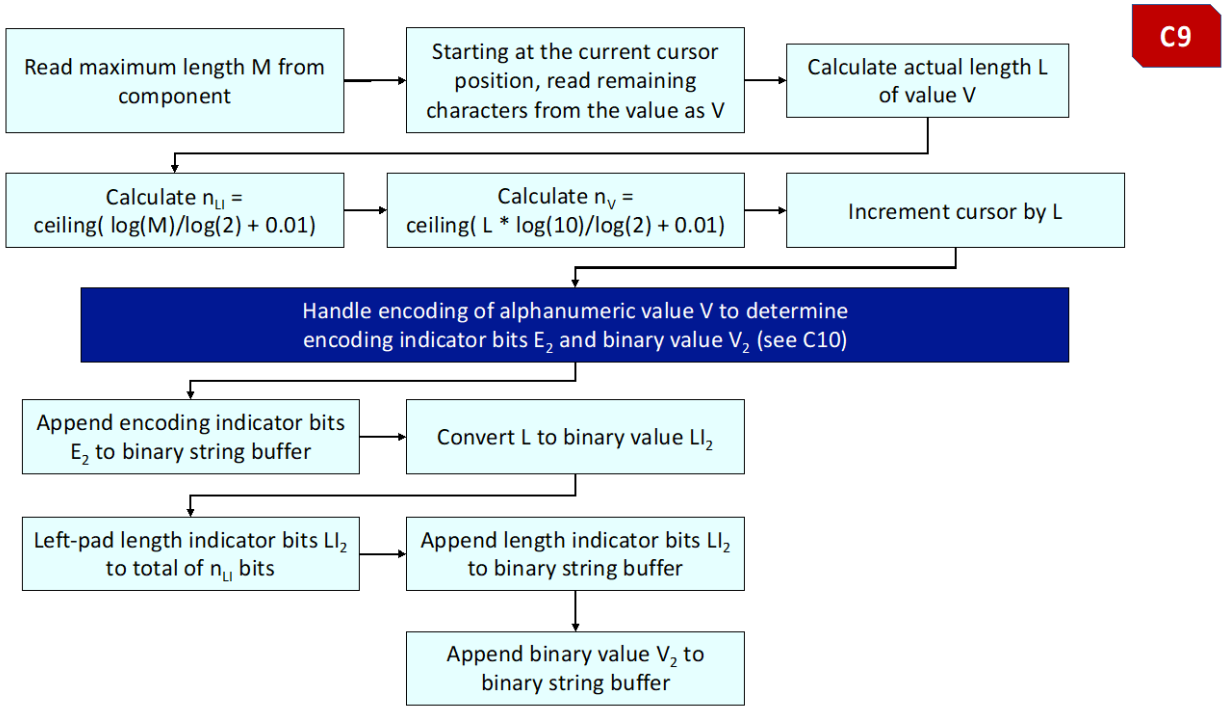
Flowchart C7 explains how to encode a variable-length numeric value. Firstly, a length indicator of an appropriate number of bit n_{LI} is encoded, according to the formula for n_{LI}, encoding the actual length L of the value. Next, the actual value is encoded as a binary integer that is left-padded 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 C8: Encode a fixed-length alphanumeric value, referring to flowchart C10 for additional details



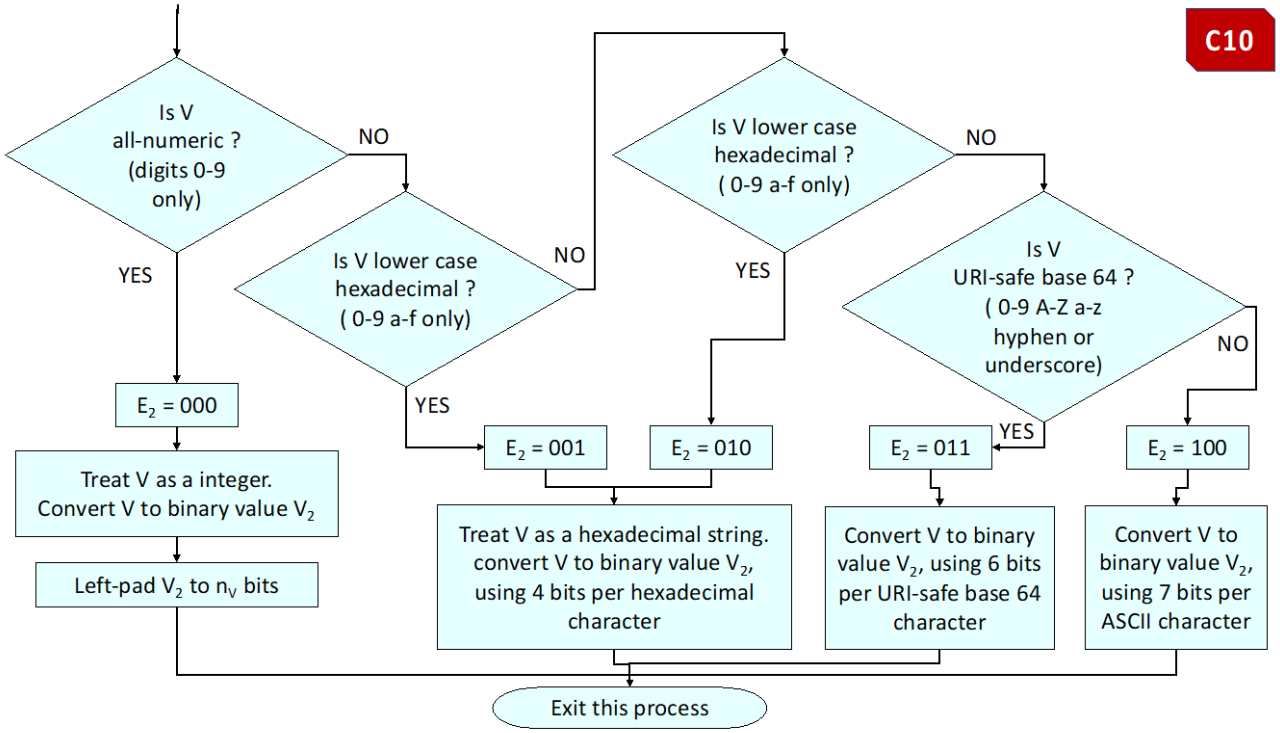
Flowchart C8 explains how to encode a fixed-length alphanumeric value. Flowchart C10 is used to determine an appropriate 3-bit encoding indicator E_2 which is encoded in the binary string before encoding the actual value binary value V_2, using an appropriate number of bits depending on its length and the encoding that was used.

Figure C9: Encode a variable-length alphanumeric value, referring to flowchart C10 for additional details



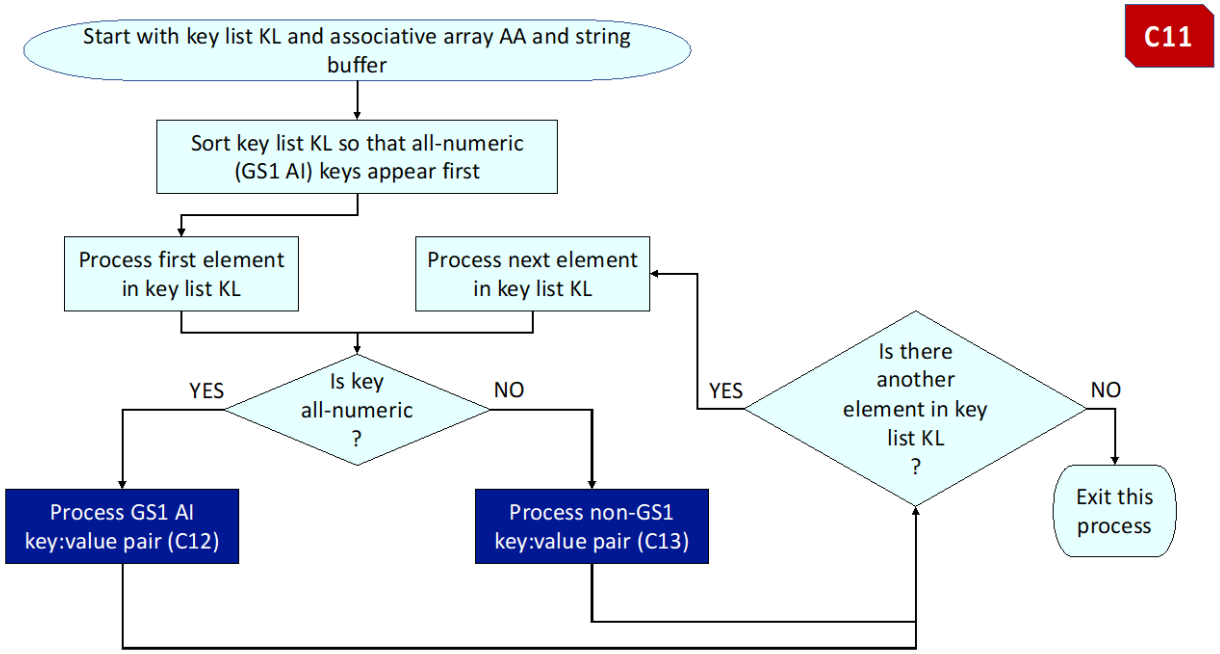
Flowchart C9 explains how to encode a variable-length alphanumeric value. This flowchart combines aspects of flowcharts C7 (variable-length, length indicator) and C8 (encoding indicator). As in Flowchart C9, Flowchart 10 is used to determine the 3-bit encoding indicator E_2 and the appropriate binary 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 C 10: Handle binary encoding of alphanumeric values



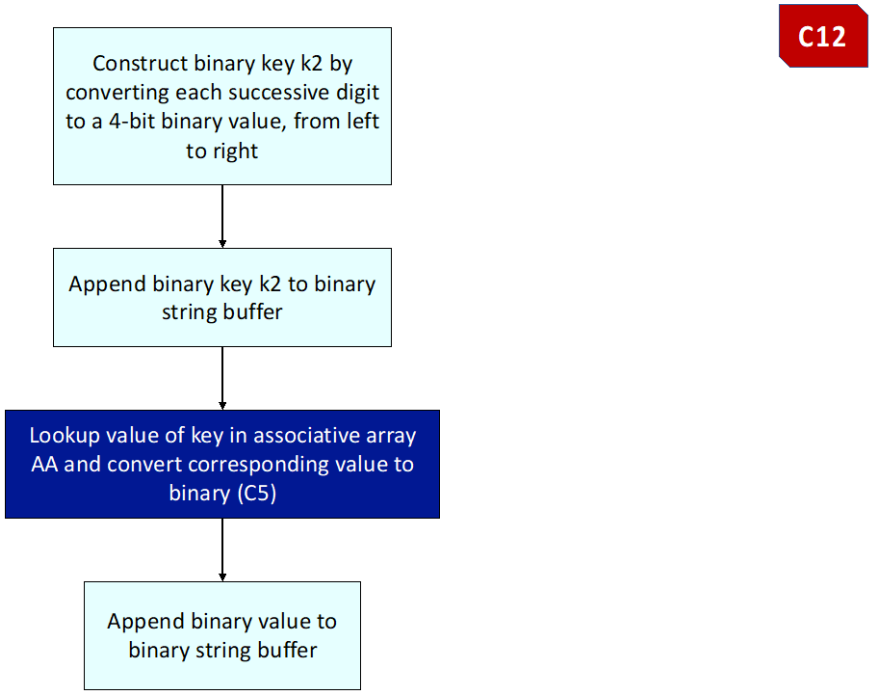
Flowchart C10 explains how to handle binary encoding of an alphanumeric value depending on whether the actual value is all-numeric, lower-case hexadecimal, upper-case hexadecimal, URI-safe base 64 or ASCII. For each of these, an appropriate 3-bit encoding indicator E2 is determined and the actual value 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 C8 or C9, since these both reference Flowchart C10.

Figure C11: Process remaining keys in the key list KL, using the associative array AA



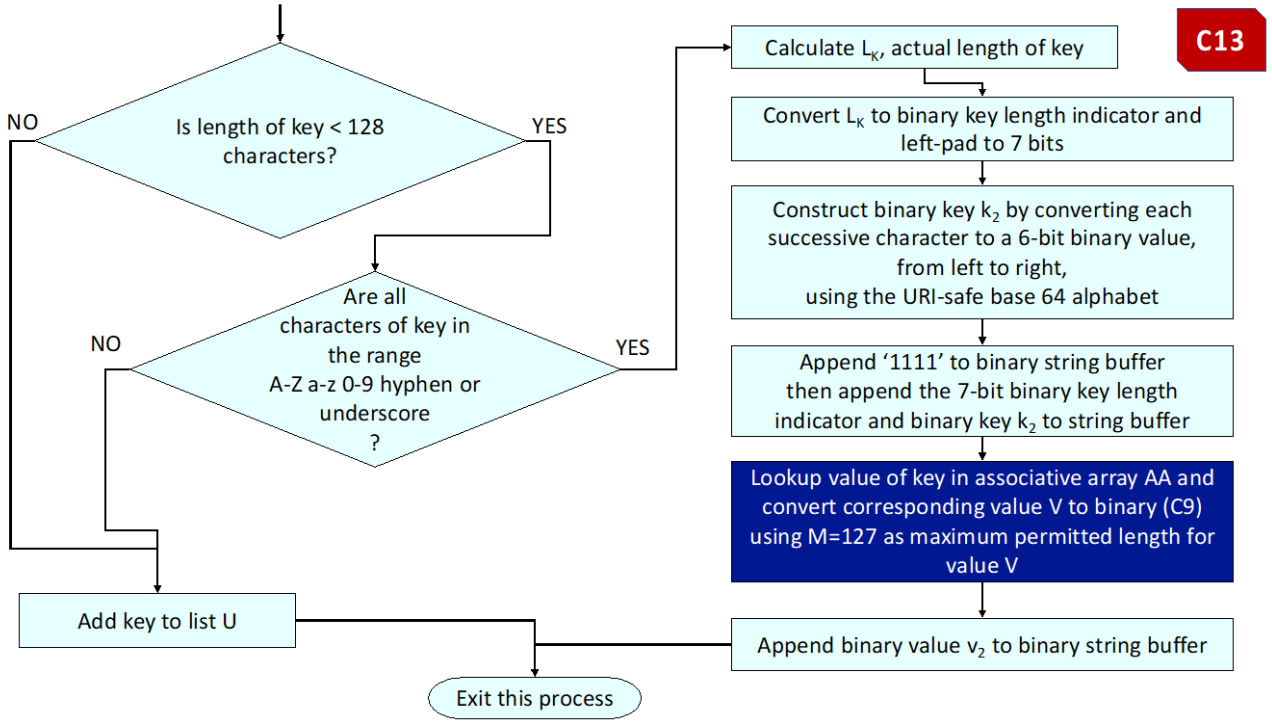
Flowchart C11 explains the processing of each key in the key list KL. The value of each key can be retrieved from the associative array AA. If the key is all-numeric, further processing references Flowchart C12 for processing of a GS1 AI key:value pair. If the key is not all-numeric, further processing references Flowchart C13 for processing of non-GS1 key:value pairs. The main loop continues to process all remaining keys in the key list KL.

Figure C12 : Encode GS1 Application Identifier key at 4 bits per digit, then encode the corresponding value in binary, referring to flowchart C5



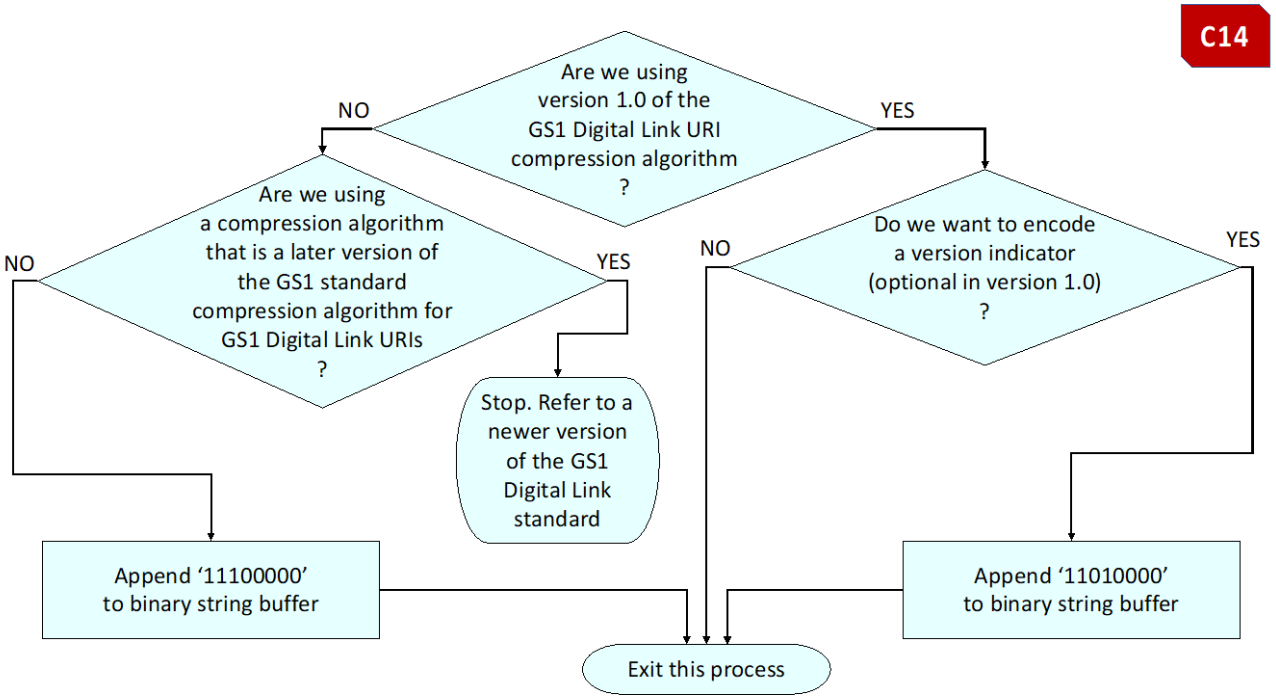
Flowchart C12 explains how to encode each GS1 Application Identifier key as a set of 4 bits per digit. Further processing then references Flowchart C5 and its dependents for formatting of the corresponding value into binary. Finally, the binary value is also appended to the binary string buffer.

Figure C13: Support compression of other non-GS1 key:value pairs from URI query string



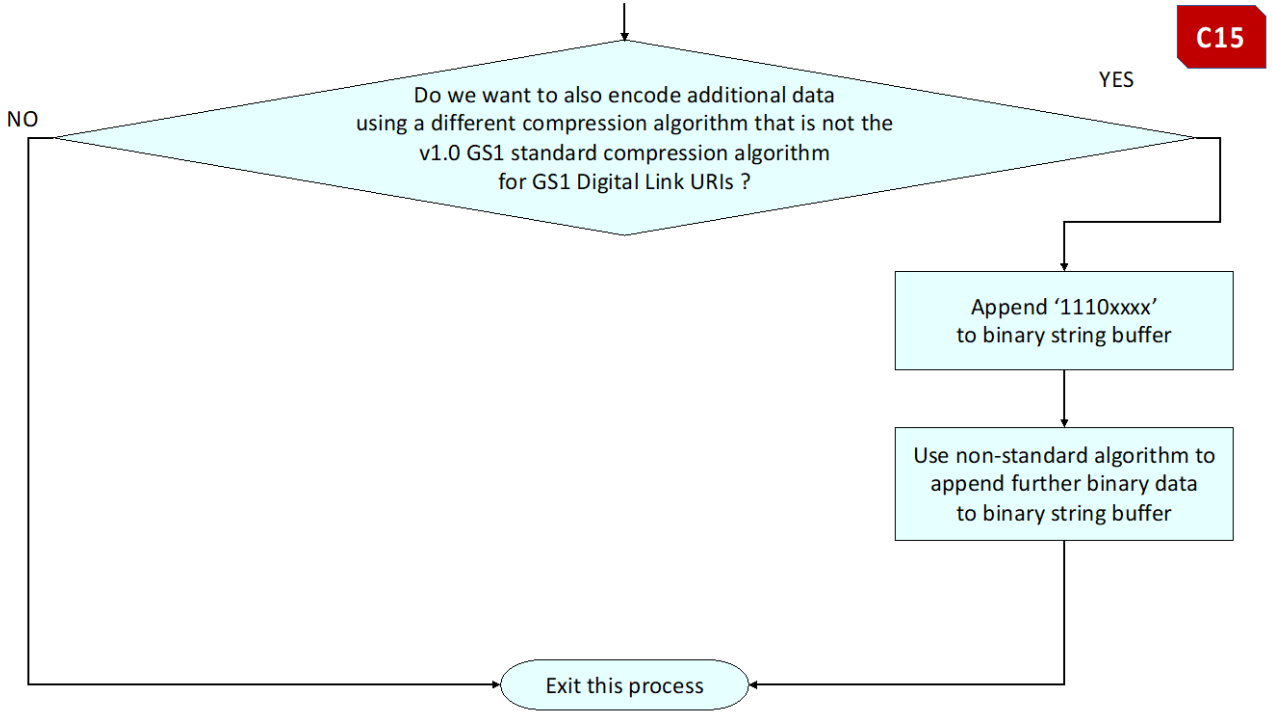
Flowchart C13 explains how to encode non-GS1 key:value pairs within the binary compressed string. For a non-GS1 key to be eligible, it must be less than 128 characters in length and all characters within the key must be within the URI-safe base 64 character set (A-Z a-z 0-9 hyphen and underscore). If 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 C9 for the formatting of the value in binary as v_2 and this is finally appended to the binary string buffer.

Figure C14: Handle version indicator (optional in v1.0)



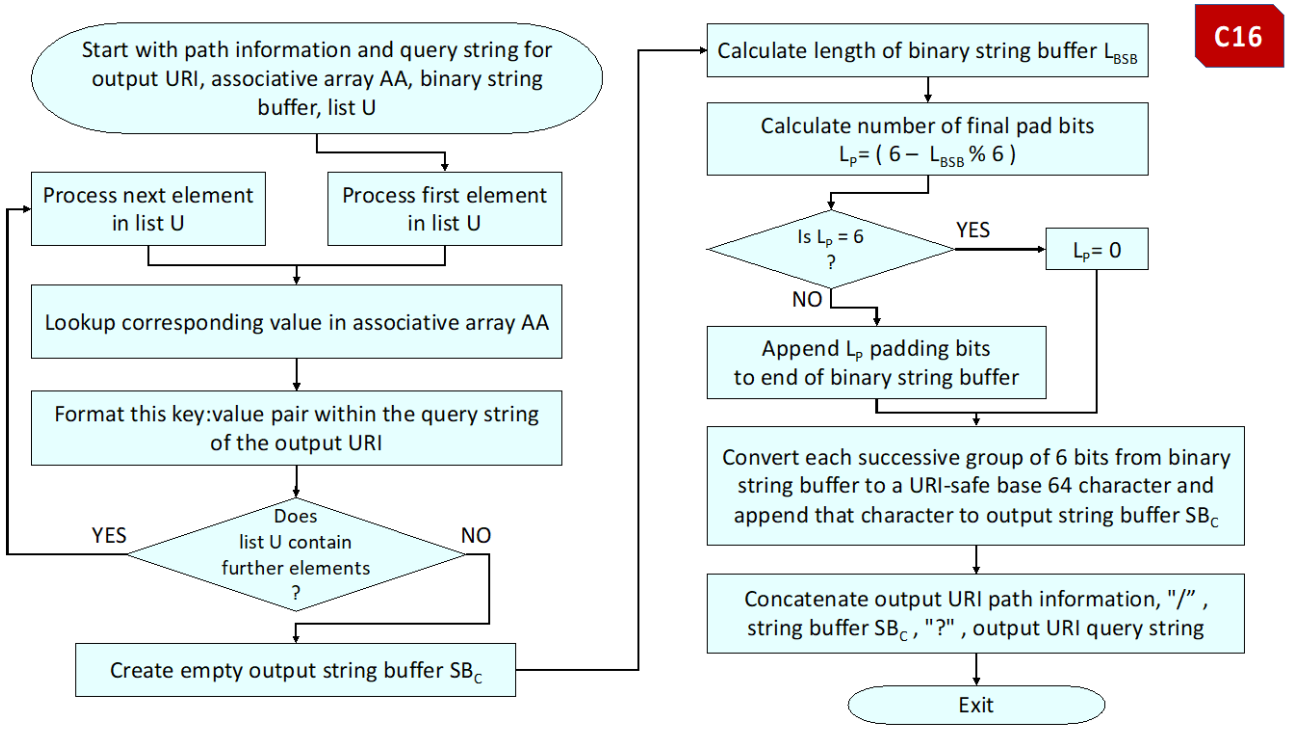
Flowchart C14 explains how to encode a version indicator. For version 1.0 of the GS1 Digital Link URI compression algorithm, the version indicator is optional but if it is encoded, it must be the value 11010000. Note that encoding a value of 11100000 (or more generally 1110xxxx) is a special reserved 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 C15: Support compression of additional data using a non-standard compression algorithm



Flowchart C15 is related to part of Flowchart C14 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 C16: Final assembly of compressed GS1 Digital Link URI

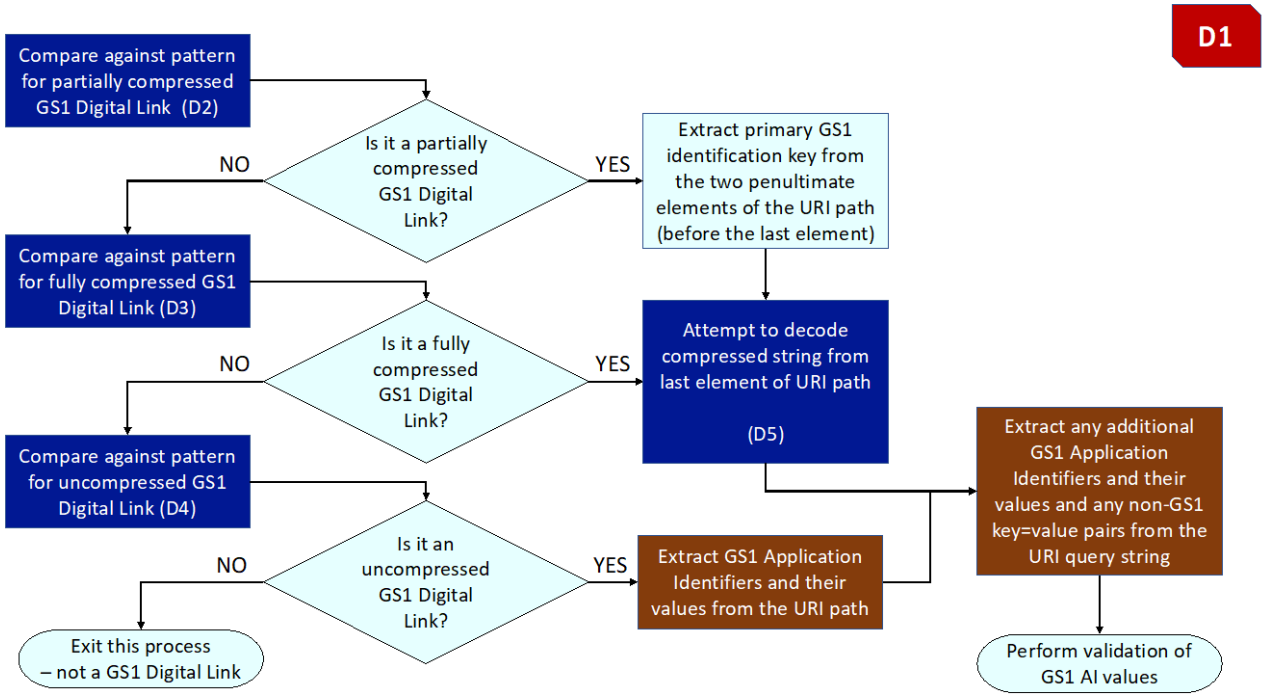


Flowchart C16 is the final high-level flowchart following on from the initial high-level Flowchart C1, which explains how to format the compressed (or partially compressed) GS1 Digital Link URI. The binary string buffer is right-padded with '0' bits to reach a total of bits that is a multiple of 6. Each 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 C1). Any remaining uncompressed key:value pairs should appear in the URI query string.

2.10 Decompression procedure and flowcharts

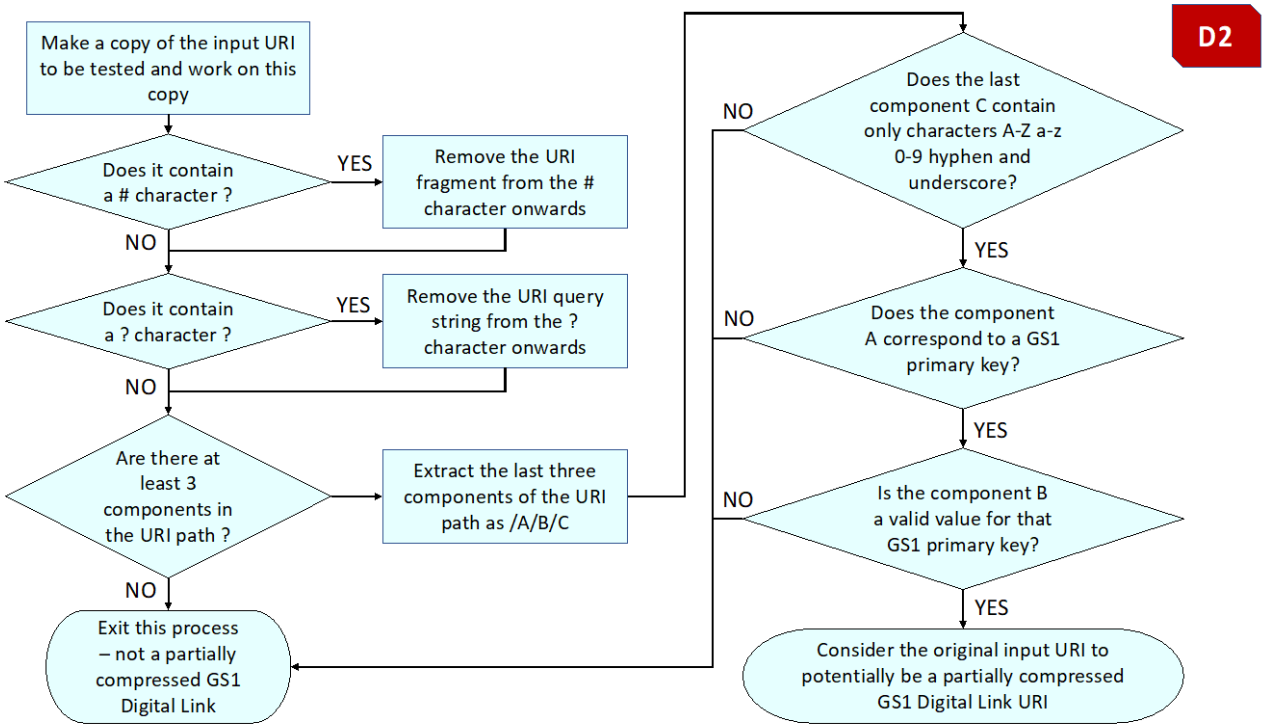
This section provides a set of flowcharts to describe the decompression procedure for fully or partially compressed GS1 Digital Link URIs. The result is an associative array of key:value pairs that can be translated to an uncompressed GS1 Digital Link URI.

Figure D1: Top-level decompression flowchart



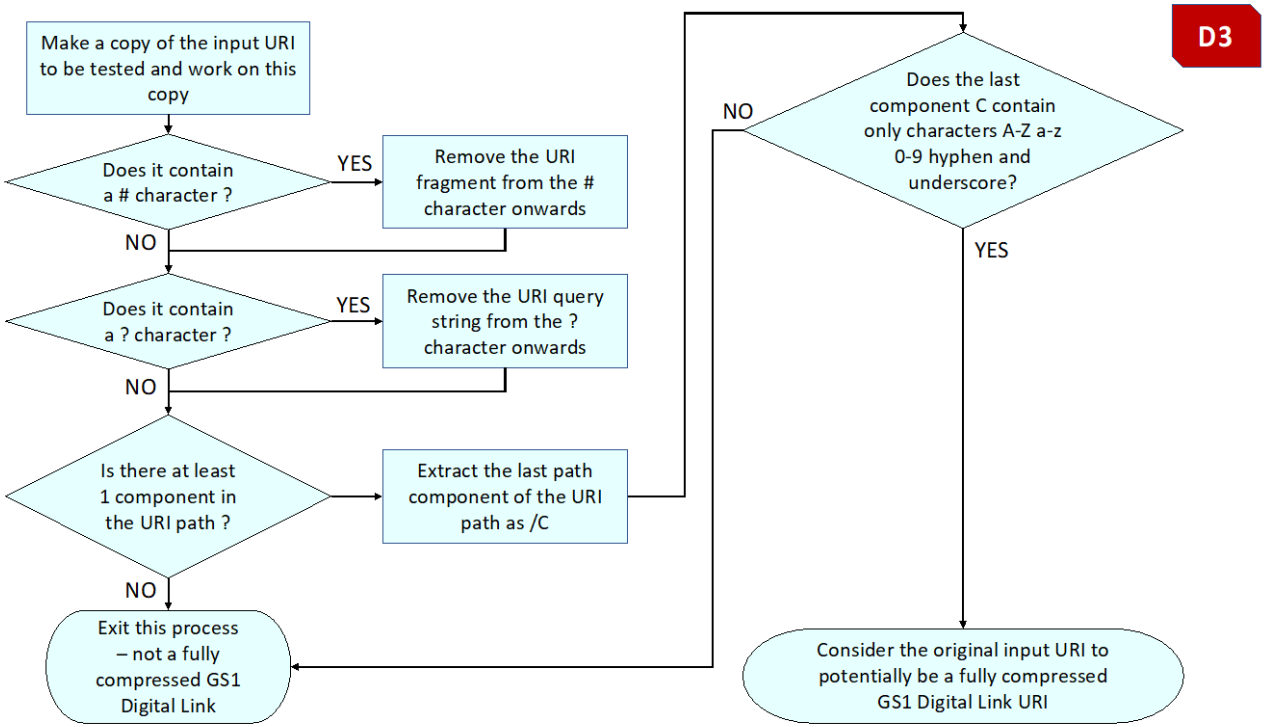
The decompression procedure begins by determining which kind of GS1 Digital Link URI is involved – uncompressed, partially compressed or fully compressed. Flowcharts D2, D3 and D4 are used to compare against patterns for partially compressed, fully compressed or uncompressed GS1 Digital Link URIs. For a partially or fully compressed GS1 Digital Link URI, flowchart D5 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.

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



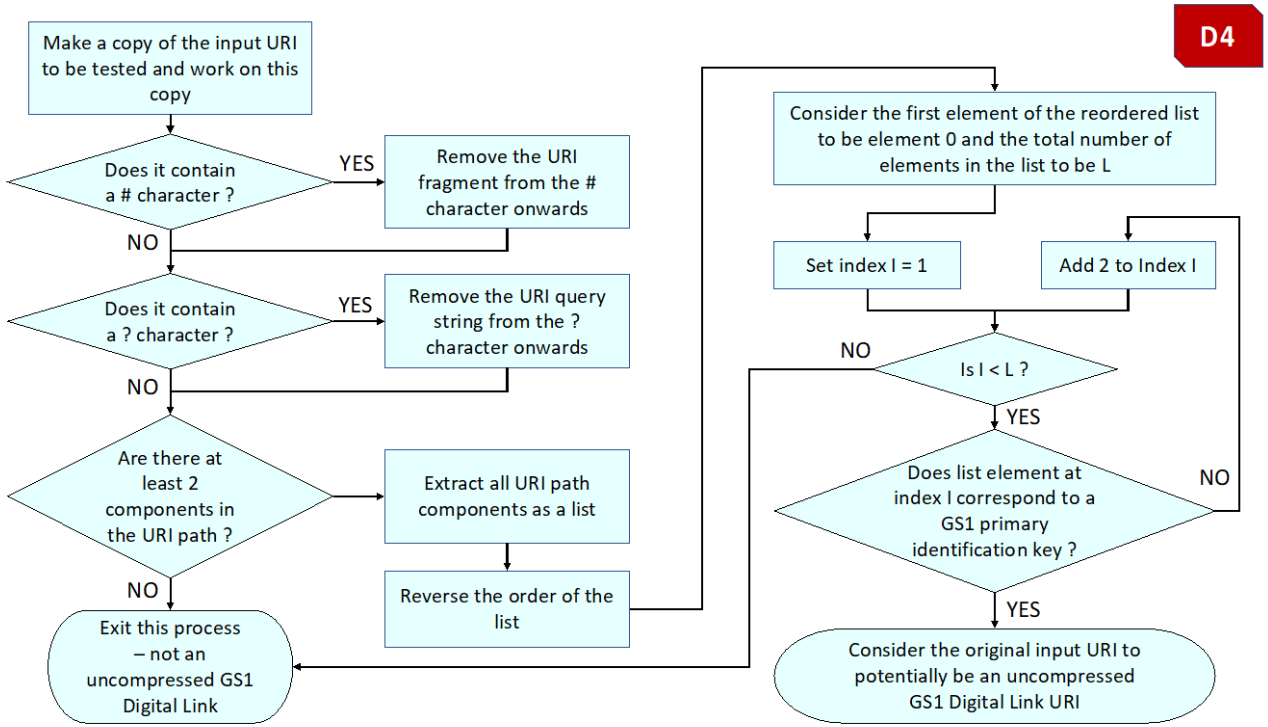
Flowchart D2 explains how to check for a partially compressed GS1 Digital Link URI in which the last component of the URI path consists of only characters from the URI-safe base 64 alphabet (A-Z a-z 0-9 hyphen and underscore) and is preceded by two URI path components, the first of which must correspond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

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



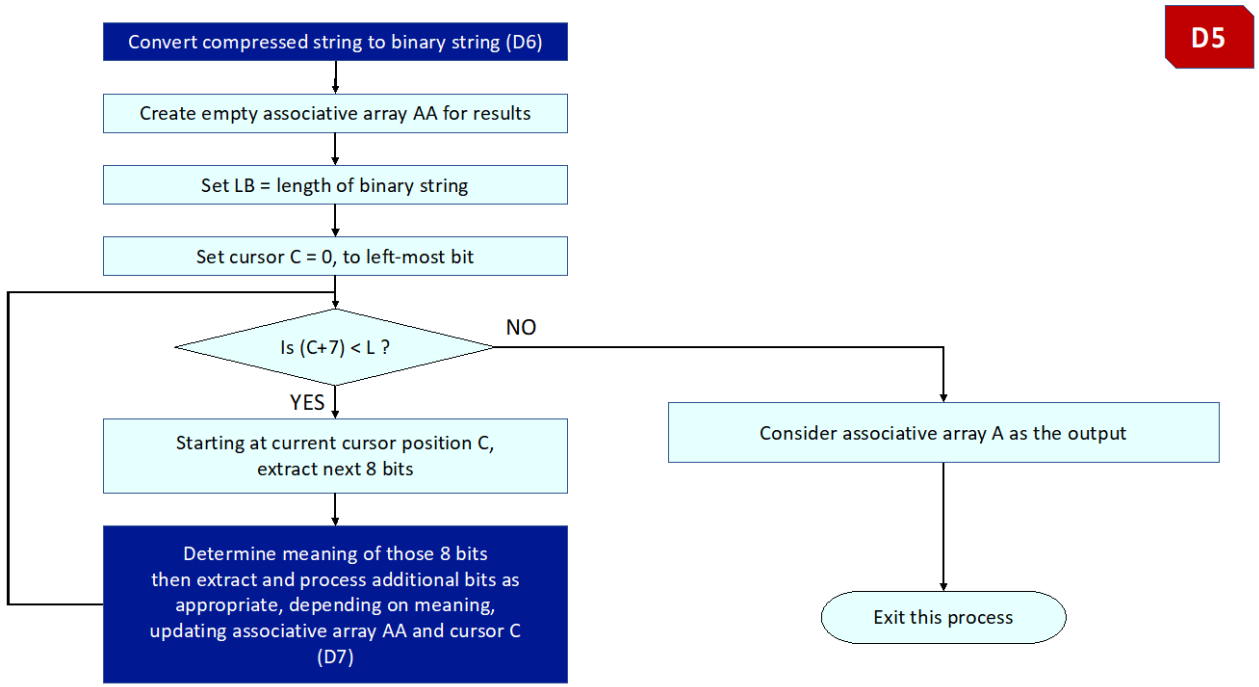
Flowchart D3 explains how to check for a partially compressed GS1 Digital Link URI in which the last component of the URI path consists of only characters from the URI-safe base 64 alphabet (A-Z a-z 0-9 hyphen and underscore). Note that because this pattern would also match a partially compressed GS1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

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



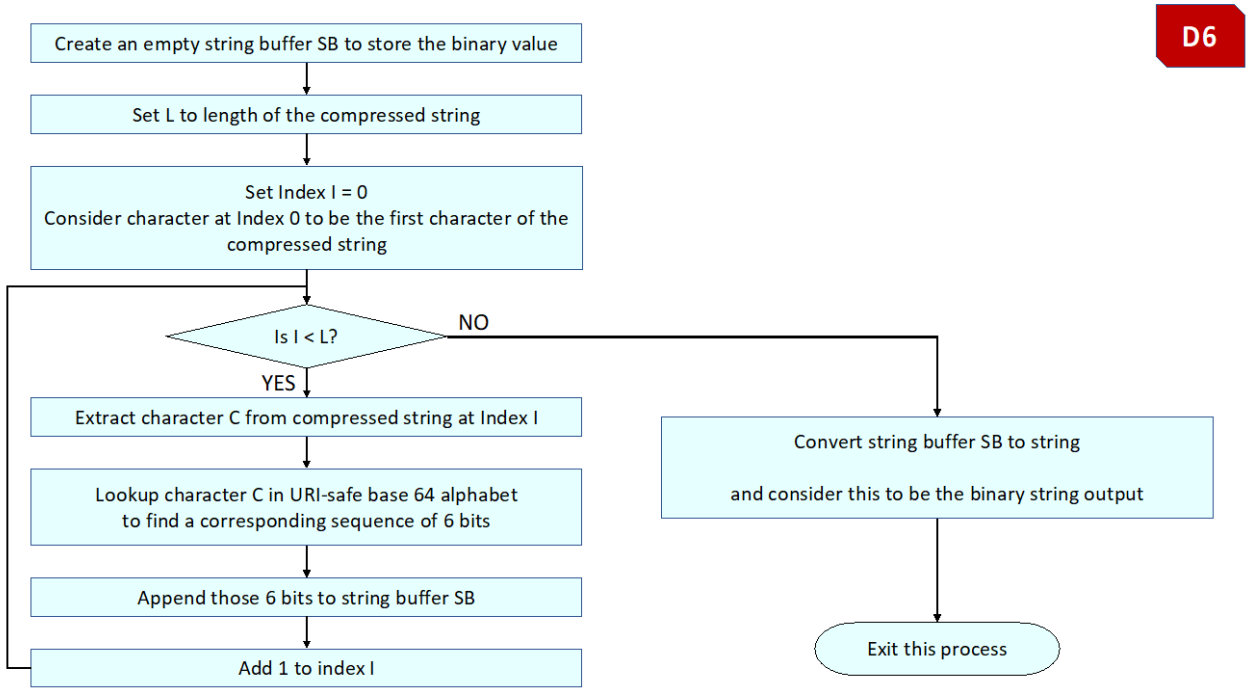
Flowchart D4 explains how to check for an uncompressed GS1 Digital Link URI. The URI path information can be analysed from right to left, considering two components at a time, testing whether the first of the pair of components corresponds to a primary GS1 identification key such as GTIN. Note that this 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 D5: High-level decoding flowchart, references flowcharts D6 (convert URI-safe base 64 characters to binary) and D7 (extract meaning and populate associative array)



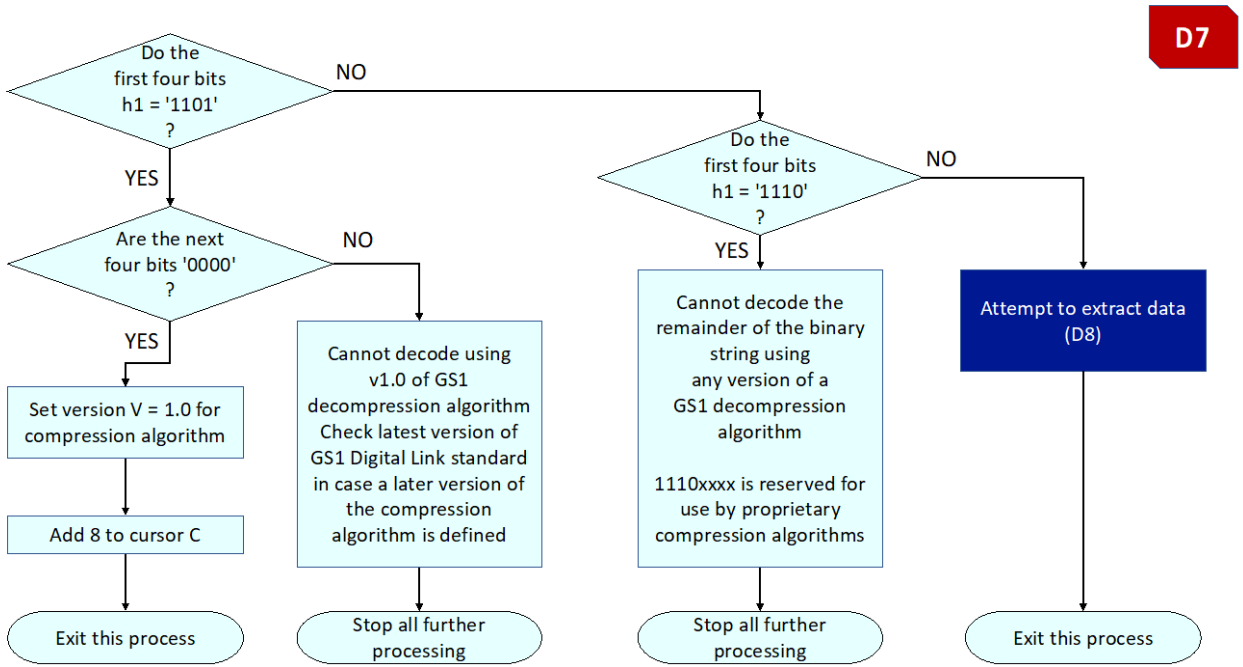
Flowchart D5 is the high-level flowchart for decoding the compressed string that is the final URI path component of a partially or fully compressed GS1 Digital Link URI. It references Flowchart D6 for the conversion from the URI-safe base 64 alphabet characters into a binary string. The main processing loop of Flowchart D5 begins by extracting 8 bits (provided that there are at least 8 bits remaining in the binary string) and then references Flowchart D7 and its dependent flowcharts to determine the meaning of those 8 bits.

Figure D6: Convert the compressed string from URI-safe base 64 characters to a binary string



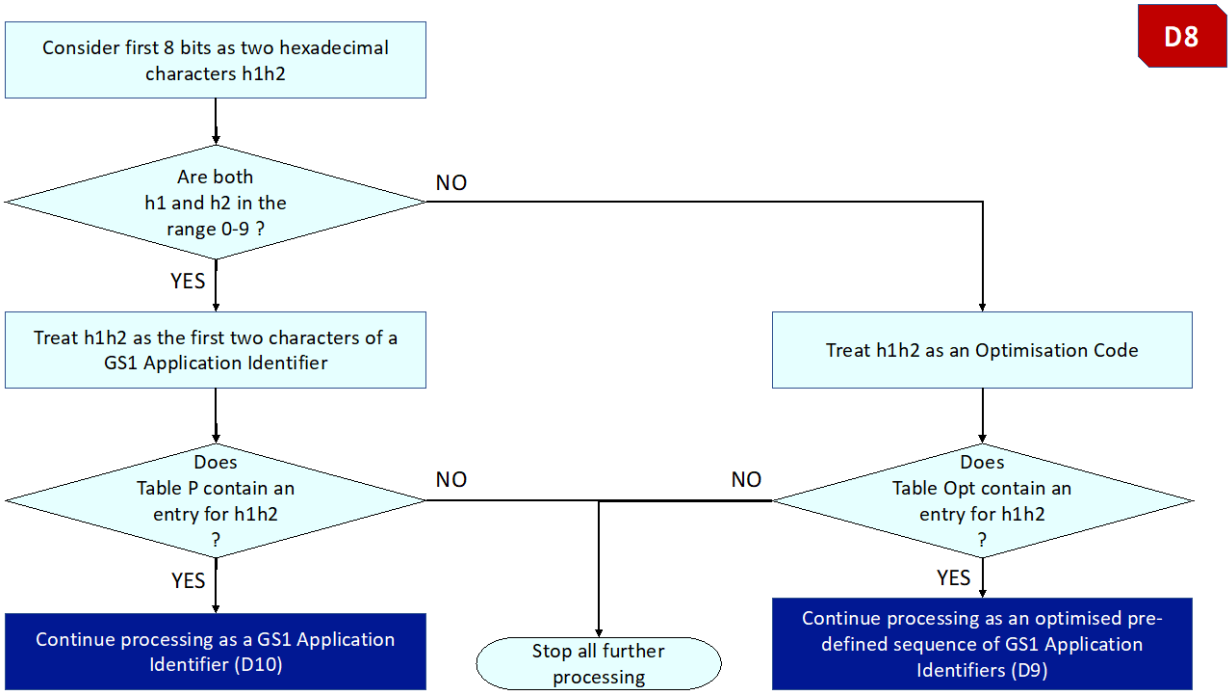
Flowchart D6 explains how to convert the compressed string expressed using URI-safe base 64 characters into a binary string. Each character encodes a sequence of 6 bits. The compressed string is decoded from left to right and the corresponding set of 6 bits per URI-safe base 64 character are also appended 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 D7: 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 D8



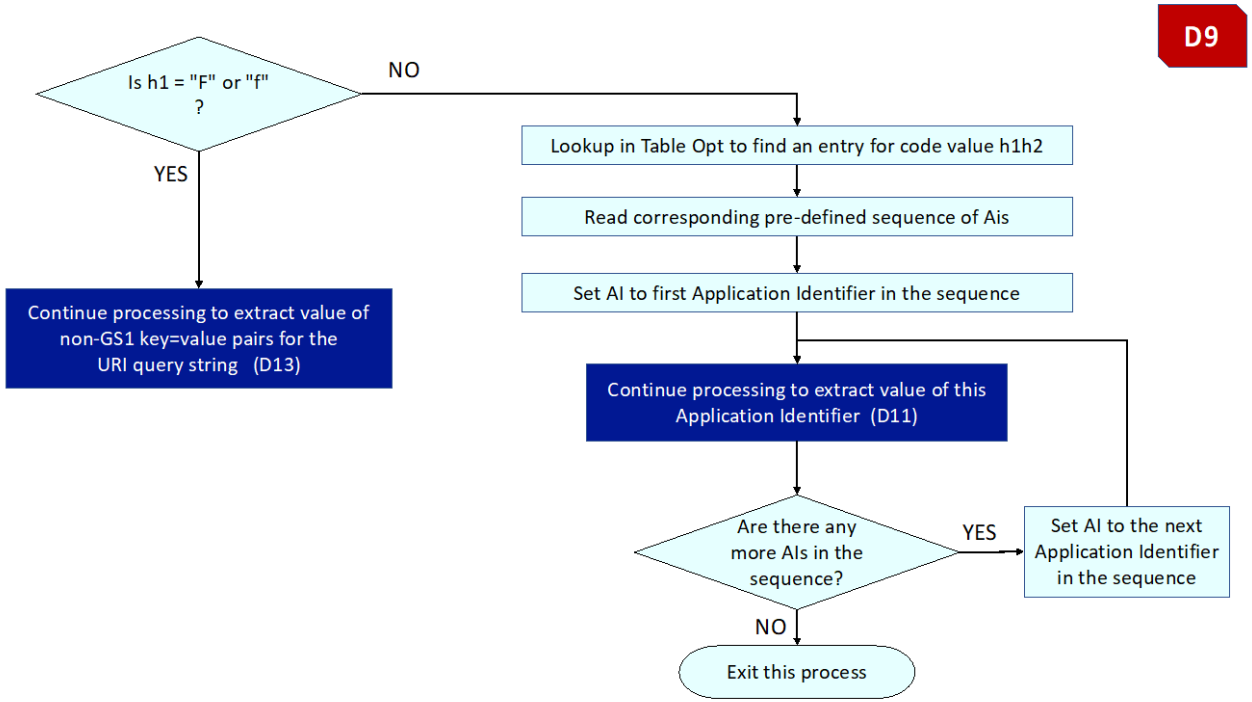
Flowchart D7 explains a sequence of checks on each 8-bit sequence read from the binary string on each iteration of the main loop of Flowchart D5. This includes detection of a potential version number. Currently only version 1.0 of the GS1 decompression algorithm is defined and identified as binary sequence 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 D7 references D8 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 D8: Extract data from binary string



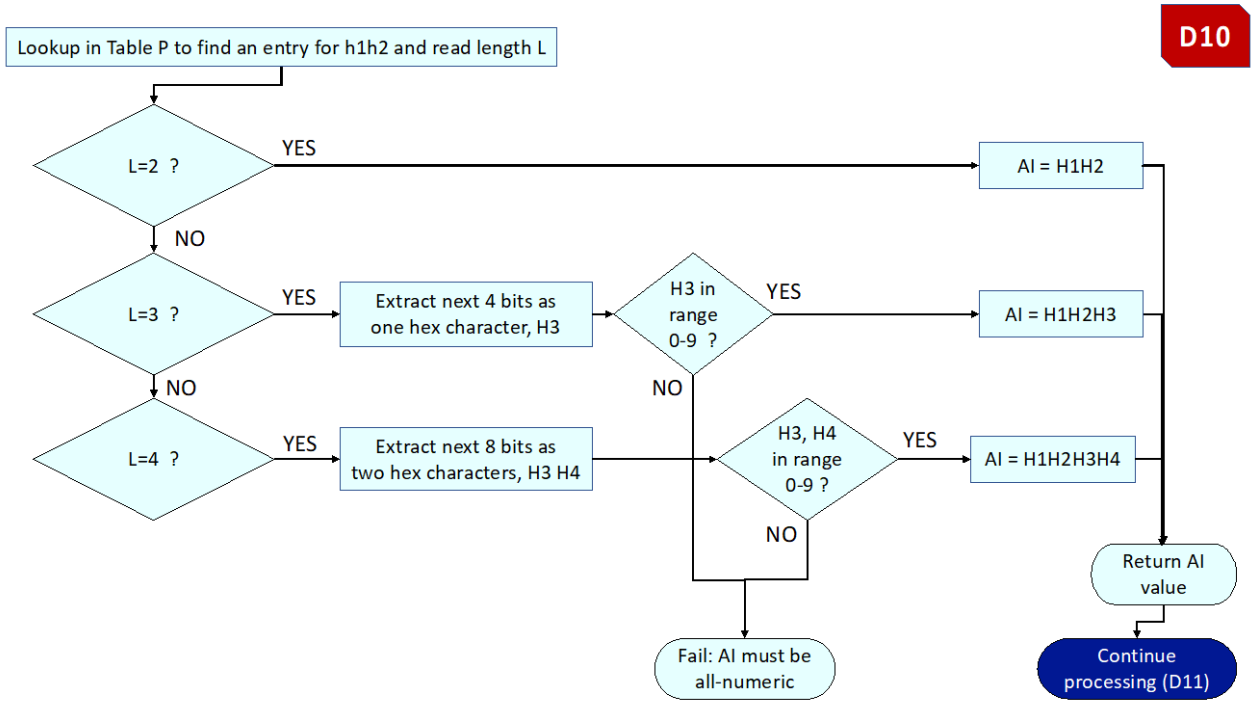
Flowchart D8 explains how to check whether the sequence of 8 bits corresponds to the first two digits of a GS1 Application Identifier of 2, 3 or 4 digits or whether it corresponds to an 8-bit optimisation code that corresponds to a predefined sequence of GS1 Application Identifiers, for more efficient 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 D10 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 D9 for handling of optimisations consisting of pre-defined sequences of GS1 Application Identifiers.

Figure D9: Processing as an optimised pre-defined sequence of GS1 Application Identifiers



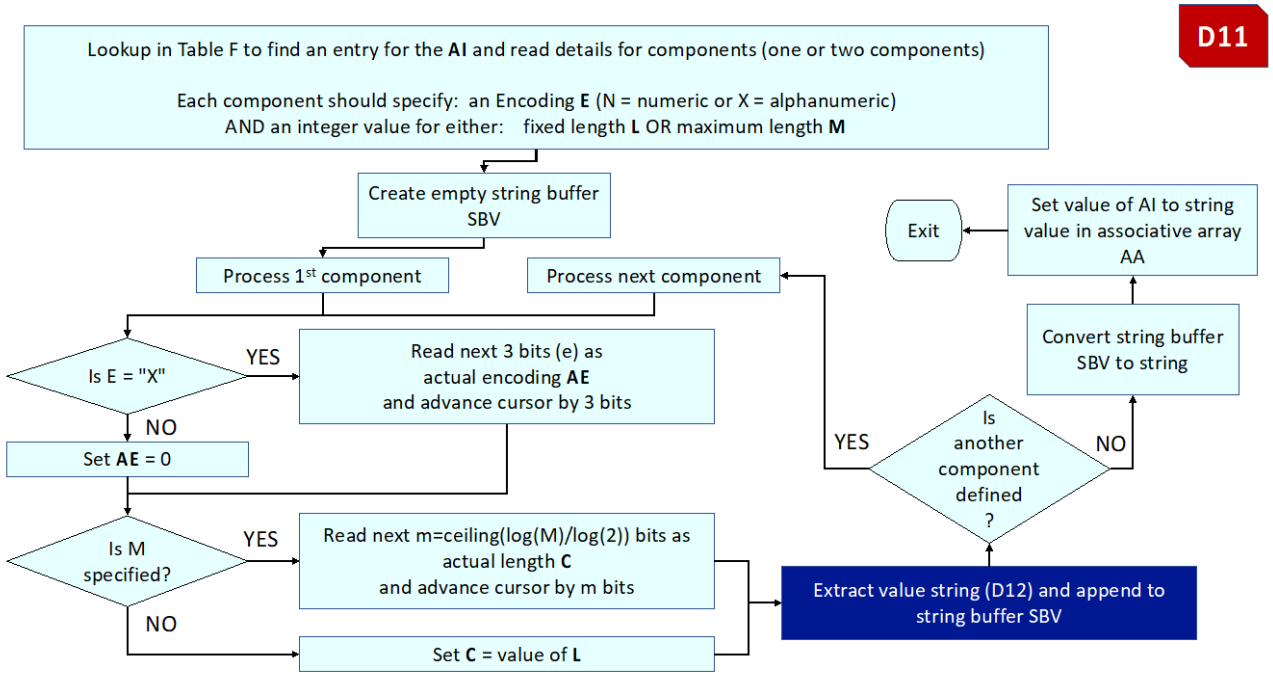
Flowchart D9 begins by performing a test whether h1 corresponds to hex character f/F (1111 in binary). If so, processing continues to Flowchart D13 because the optimisation code range Fx (1111xxxx) is reserved to support compression of non-GS1 key:value pairs from the uncompressed URI query string into the compression string. If h1 is not hexadecimal character f/F, then 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 D9, referencing Flowchart D11 for extraction of the value for each GS1 Application Identifier.

Figure D10: Process as a single GS1 Application Identifier



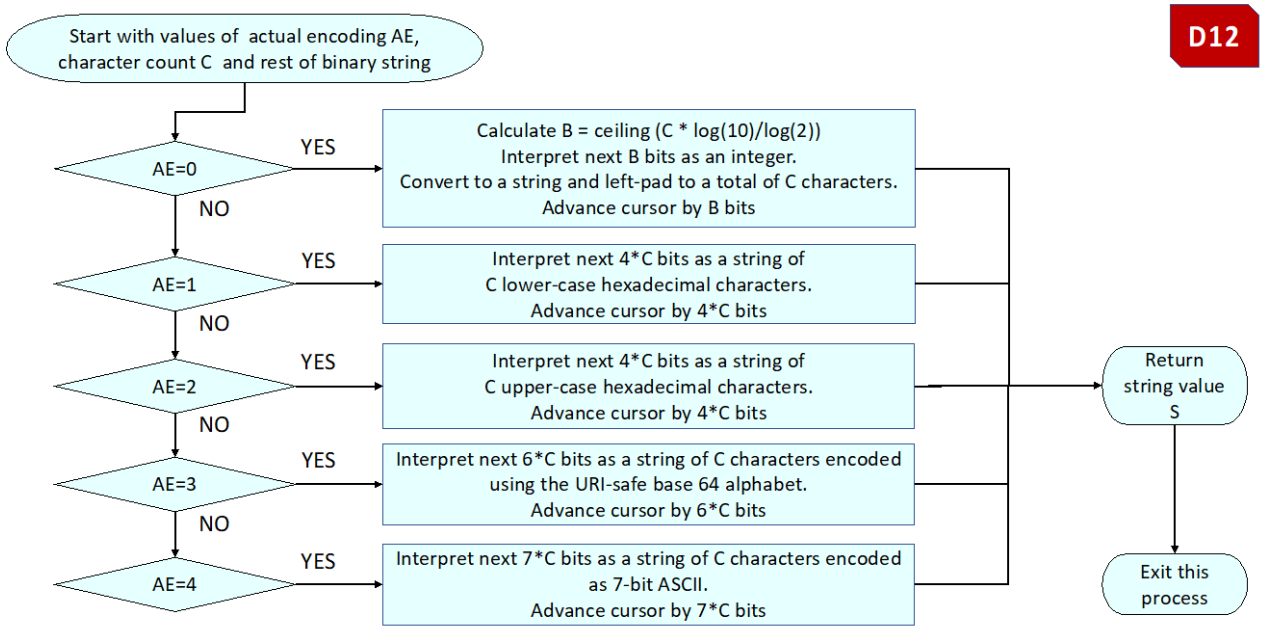
Flowchart D10 is used to determine whether the initial digits h1h2 correspond to a 2-digit, 3-digit or 4-digit GS1 Application Identifier. Table P contains the current rules for making this determination based on the initial two digits. If Table P indicates a 3-digit GS1 Application Identifier (e.g. 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 D11 to extract the value for the GS1 Application Identifier key identified in this flowchart.

Figure D11: Determine values for actual encoding AE and character count C for each component



Flowchart D11 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 D12 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 D12: Extract the value from the binary string



Flowchart D12 explains how to extract an appropriate number of bits for a specific value of actual encoding AE and character count C (determined in Flowchart D11). 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 D13: Extract the value for key=value pairs

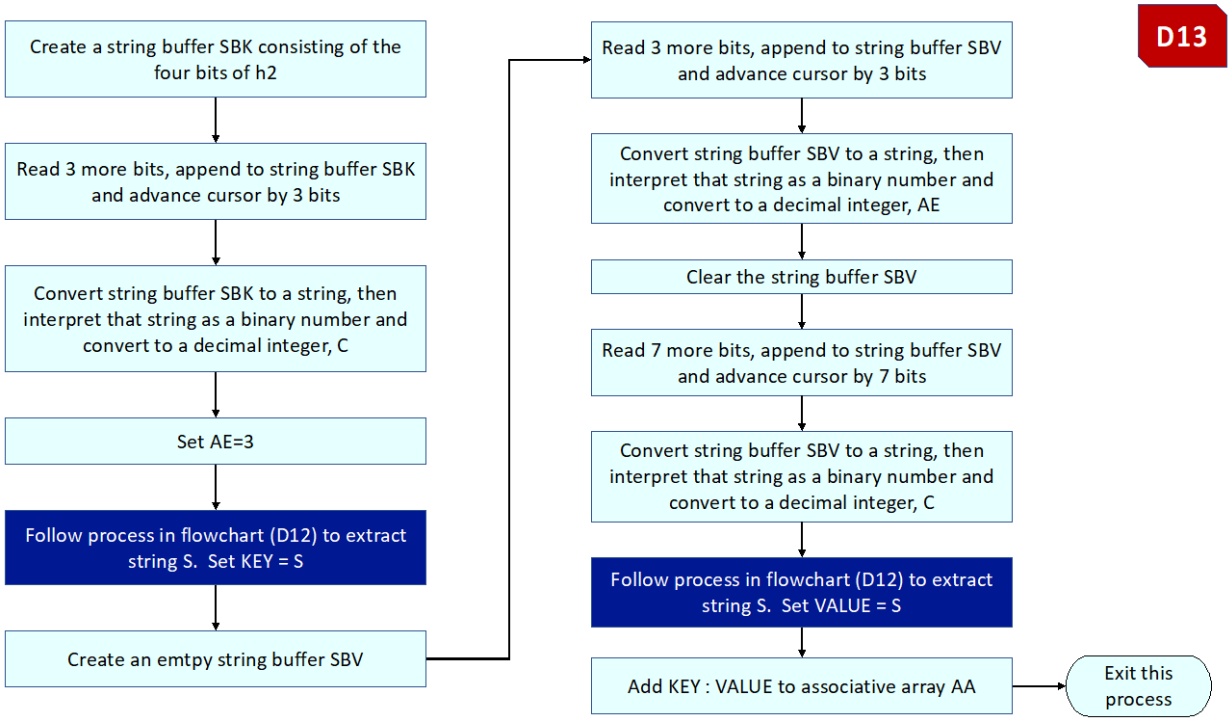


Figure D13 explains how to extract non-GS1 key:value pairs from a compressed string; after decompression these are restored to the URI query string. The four bits of h2 and a further 3 bits are interpreted as a 7-bit length indicator, permitting a key up to 127 characters. The key is always encoded using 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 D12 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.

3 Semantics

This section and all its subsections are normative unless flagged otherwise.

At the time of writing, the use of terms from the schema.org vocabulary is under review and subject to change in a future version of this standard.

The following prefixes are used in thus chapter.

Table 3-1 Prefixes and namespaces used in this document

Prefix	Namespace
gs1	https://ref.gs1.org/voc/
schema	http://schema.org/
dcterms	http://purl.org/dcterms/
skos	http://www.w3.org/2004/02/skos/core#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#

3.1 Exposing GS1 Digital Link semantics to the outside world

This subsection is informative

The GS1 community is very familiar with the meaning of things like GTINs, SSCCs, expiry dates and so on. However, there is a great deal of background knowledge that a human uses to interpret the GS1 system. GS1 Digital Link operates within and outside the GS1 community and so needs to be much more precise if information is to be exchanged accurately between non-specialist people and, even more critically in this context, non-specialist information systems. Therefore we must add precise detail concerning what is expressed using GS1 Digital Link URIs, the relationships between related GS1 Digital Link URIs and also how we can express facts about things that are identified using them, not only as human-readable Web pages that include tables of information, but also as structured data that can be automatically interpreted by computer software.

The GS1 Digital Link URI syntax is just a way of expressing a set of one or more GS1 element strings in a Web-friendly format that looks like a Web address or URL and functions as a Web address, in the sense that a Web request (HTTP / HTTPS GET request) can return relevant data.

GS1 Digital Link supports all fundamental GS1 identification keys (e.g. GTIN, SSCC, GRAI, GIAI, GSRN etc.) as well as appropriate qualifiers (e.g., Consumer Product Variant, Batch/Lot Number, Serial Number, GLN extension, CPID Serial Number) that can be used to form compound identification keys that enable identification at a finer granularity, such as a specific batch of products or even an individual product instance. Additionally, the GS1 Digital Link URI syntax also supports the expression of data attributes for which corresponding GS1 Application Identifiers are defined. These include data attributes such as net weight, gross weight, expiry date, date of production, dimensions, country of origin, etc.

The good news is that Semantic Web/Linked Data technology provides an effective, standardised way to exchange factual data about data attributes and their values without the need to exchange long GS1 Digital Link URIs. This chapter explains how that technology works and how it can be used to exchange factual data about things so that the meaning can be automatically interpreted by computer software within and beyond the GS1 information ecosystem.

3.2 Equivalence

The GS1 Digital Link URI syntax has been developed with the practical realities of the supply chain and commercial worlds very much in mind. This is the thinking behind some key features, notably:

- domain name neutrality (you can create a conformant GS1 Digital Link URI using any domain name);
- the definition of developer-friendly short alpha codes, like 'gtin' and 'cpv' as equivalents of their numeric GS1 application identifiers, so that:
  `https://example.com/foo/gtin/614141123452/cpv/2A`
  `https://id.example.org/gtin/614141123452/cpv/2A`
  `https://example.example/01/614141123452/22/2A`
  `https://id.gs1.org/01/614141123452/cpv/2A`
  `https://id.gs1.org/gtin/614141123452/22/2A`
  are all equivalent in terms of the information they carry and the items they identify.

Fulfilling these needs comes at a cost since it is clear that there can be an infinite number of conformant GS1 Digital Link URIs that identify the same thing. Whilst this does not go against the Architecture of the World Wide Web [WebArch], it does mean that further clarity is required so that conformant GS1 Digital Link URIs can be used with confidence in Semantic Web and Linked Data environments.

Non-unique naming is commonplace in Linked Data but the fact that two URIs identify the same resource will often need to be recognised, and in some cases explicitly declared, in information processing systems. In this context, it is worth noting two similar but distinct relationships that occur commonly.

Starting from a GS1 Digital Link URI:

- The correct relationship with another URI that identifies the same item, or class of items, is `owl:sameAs`, the definition of which is "indicates that two URI references actually refer to the same thing: the individuals have the same *identity*". This relationship type is very specific and should be used with care. It is, however, the correct relationship between the URIs listed above.
- It is anticipated that the more common case will be a link to information describing the item, particularly a product information page. That is, the identified item and the description of it are two distinct concepts and therefore `owl:sameAs` is not applicable. The correct relationship in this case is likely to be `gs1:pip`, `gs1:smpc` etc.

See the GS1-Conformant Resolver Standard for more on relationship types [Resolver].

3.3 Information encoded within the URI

There is a feature of the GS1 Digital Link URI syntax that goes directly against W3C's Web architecture principles [WebArch], namely URI Opacity which states that:

> *Agents making use of URIs SHOULD NOT attempt to infer properties of the referenced resource.*

GS1 recognises this conflict, however, it is overridden by the need to meet two requirements:

- that a Web address directly related to a product can be encoded in its entirety in a data carrier on the pack (in a QR Code®, NFC tag, digital watermark or similar) such that a generic application with no special software can directly extract the entire URL and follow the link, without the need to further construct the URL;
- that critical applications within the supply chain do not need to dereference the GS1 Digital Link URI in real time to extract the GS1 keys and key values; they can instead extract the GS1 keys and key values using translation software that requires no real-time online connectivity.

GS1 further notes the definition of SHOULD NOT from [RFC 2119]:

> *... there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.*

With these factors in mind, it is important to note that the semantics of a GS1 Digital Link URI can only be inferred within the specific context of a GS1-aware system. This means that, in addition to normal error handling, any application attempting to infer semantics from a GS1 Digital Link URI SHALL first check that:

- it is conformant to the standard, including validity of the keys and key values;
- where multiple keys are included in the URI that the combination is itself valid, see section 4.14 of the GS1 general Specifications [GENSPECS].

Furthermore, the development of a URI syntax that is independent of any internet domain name goes directly against *URI Design and Ownership* [BCP190], the abstract of which includes:

> *… the structure of a URI is defined by its scheme. While it is common for schemes to further delegate their substructure to the URI's owner, publishing independent standards that mandate particular forms of URI substructure is inappropriate, because that essentially usurps ownership.*

It is recognised that GS1 Digital Link goes against this by defining a URI substructure but notes that this standard *requires* that all conformant resolvers SHALL publish a resolver description file at /.well-known/gs1resolver that asserts its independent control of its URI space. This includes information about whether the resolver handles all or only a subset of the GS1 identifier system, support for any extensions, the supported value space(s), if any, for the context keyword and more.

Therefore, each resolver/internet domain remains sovereign over its URI space and the BCP is not contravened.

3.4 Trailing slashes

Where a GS1 Digital Link URI resolves to a directory on a server, it will typically be appended with a trailing slash automatically. Thus our example

`https://example.com/gtin/614141123452/cpv/2A`

might be seen in the address bar of a browser as

`https://example.com/gtin/614141123452/cpv/2A/`

This is **not** conformant, however, it is likely to be common and therefore resolvers SHOULD be tolerant of it.

3.5 The identified resource and the applicability of attributes

Historically at least, the basis of the GS1 System is the provision of identifiers for physical things. Therefore, GS1 Digital Link URIs also identify physical things, not information resources that describe them. Making that statement does, inevitably, lead to the potential problem of dereferencing the URI for a physical product to discover the obvious nonsense that it has a media type of text/html and a size of 13.4 kilobytes. This is a very old and well known problem but it is one that presents few practical problems in the real world where the assumption that the identifier is for the physical object is usually safe.

Nevertheless, it is important to think carefully about what we really mean by identifiers - and to carefully distinguish between a class of objects and an individual instance within that class.

The GS1 Glossary defines a class as "A class describes a set of objects that share the same attributes, relationships and semantics." The GS1 Glossary does not currently provide a definition for instance, nor does the GS1 System Architecture [Arch] - although the concept of instance-level identification and instance-level data is discussed throughout the GS1 System Architecture document, e.g. in sections 6.1.1.1 and 6.2.

We are familiar with products that carry a GTIN barcode. In everyday language, we might say that the GTIN identifies the product. However, we also recognise that in practice there are multiple copies (instances) of the same mass-produced product, each sharing the same GTIN barcode.

The GTIN is not sufficiently specific to identify each individual product instance; for that we need to combine the GTIN with a Serial Number that is unique within the GTIN class and which is different for each instance, i.e. no two instances of the same product would be allowed to have the same combination of GTIN and Serial Number. This combination of GTIN + Serial Number is noted in the GS1 System Architecture, see Table 4-1 within section 4.2.

We therefore conclude that the familiar GTIN barcode identifies the product class, rather than the individual product instance.

The GTIN is still useful for retrieving product master data defined for that product class, such as the ingredients or material composition, net weight etc. However, other data, such as traceability data, physical event data or transactional data may be concerned with that individual product instance, such as its date of manufacture, date of expiry or date of purchase.

In situations where the individual product instance is not uniquely identified, we may still want to express factual statements such as:

- 'this instance of the product with GTIN 01234567890128 was manufactured / packed on date 2018-04-02'
- 'this instance of the product with GTIN 01234567890128 was sold on date 2018-06-07'
- 'this instance of the product with GTIN 01234567890128 has an expiry date 2018-07-21'

We also understand that these factual statements only apply to *this instance* of the product, not to all replica instances that share the same GTIN'.

We may want to express such factual statements in a machine-interpretable way using Linked Data technology. In situations where the instance of a product does not have a data carrier that identifies its GTIN and its Serial Number, then Linked Data technology provides the concept of a 'blank node', which simply means 'this thing with no globally-unambiguous URI name'.

A blank node identifier is often written as _:1, _:2 etc., where underscore (_) just indicates a local namespace with no specific mapping to a global URI. Blank node identifiers are useful because we can write multiple factual statements that share the same Subject or Value or where the Value within one statement is the Subject of another statement, even when no globally-unambiguous URI is available for these things.

In the examples above, we could write (in Turtle syntax):

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .
@prefix gs1: <https://gs1.org/voc/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

_:1  rdf:type schema:Product .
_:1  gs1:gtin  "01234567890128" .
_:1  gs1:productionDate  "2018-04-02"^^xsd:date .
_:1  gs1:expirationDate  "2018-07-21"^^xsd:date .
```

In JSON-LD, this looks like:

```
{
```

```
  "@context": {
    "gs1": "https://gs1.org/voc/",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "xsd": "http://www.w3.org/2001/XMLSchema#"
  },
  "@id": "_:f2e87b45989e249c0873cfe3aa6b79948b1",
  "@type": "schema:Product",
  "gs1:expirationDate": {
    "@type": "xsd:date",
    "@value": "2018-07-21"
  },
  "gs1:productionDate ": {
    "@type": "xsd:date",
    "@value": "2018-04-02"
  },
  "gs1:gtin": "01234567890128"
}
```

In plain English, these statements express the following:

- 'this thing is a product'
- 'this thing has a GTIN value 01234567890128'
- 'this thing was manufactured on 2nd April 2018'
- 'this thing has an expiry date of 21st July 2018'

Note that the GTIN 01234567890128 is not the Subject of these factual assertions.

The reason is that we do NOT want to say the following:

- 'EVERY instance of the product with GTIN value 01234567890128 was manufactured on 2nd April 2018'
- 'EVERY instance of the product with GTIN value 01234567890128 was purchased on 7th June 2018'
- 'EVERY instance of the product with GTIN value 01234567890128 expired on 21st July 2018'

By using a blank node to make statements about 'this thing', we avoid making invalid statements that would otherwise apply too broadly to all instances of the same product.

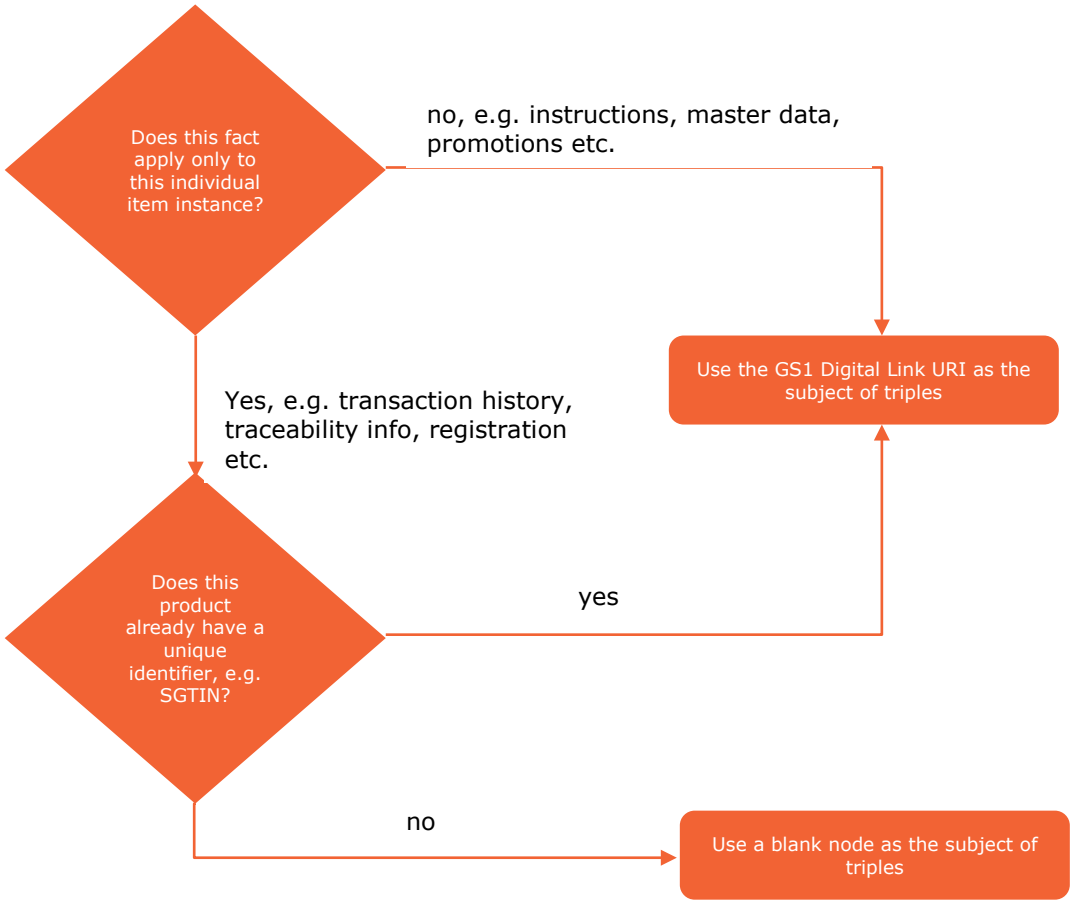
When a GS1 Digital Link URI identifies a maximum of one object in the world, e.g. a unique combination of GTIN and Serial Number (or other individual instance identifiers for things that are not products), then it is correct to use the GS1 Digital Link URI as the subject of factual statements that are specific to the instance (such as any facts relating to the traceability or transaction history of the individual instance).

It is also correct to use the GS1 Digital Link URI as the Subject of facts that are defined at class-level (GTIN granularity) or sub-class granularity (e.g. GTIN+Batch/Lot or GTIN+CPV) and always apply to all instances within that class or subclass.

If the individual object does not carry a globally unique instance identifier (GTIN + Serial Number unique within that GTIN), then Linked Data applications SHALL instead use a blank node as the

subject of any factual statements about the traceability data or transaction history of that specific instance of a product.

Figure 3-1 Decision tree for whether to use the GS1 Digital Link URI or a blank node as the subject of triples expressing facts about the identified item



The flowchart above attempts to provide some guidance about when to use the GS1 Digital Link URI as the subject of Linked Data triples (factual assertions) and when it is more appropriate to instead use a blank node that has no globally unambiguous URI name - only a local name for combining related facts and referencing them from other facts.

Table 4-1 of the GS1 System Architecture document [Arch] identifies the following compound keys that identify individual instances of things. That table is repeated below for convenience.

Table 3-2 Combinations of common AIs that identify instance-level items

Entity	Physical / Digital / Abstract	Candidate key
Trade Item Instance (Product Instance)	Physical or Digital	GTIN + AI 21 (compound) = 01 & 21
Returnable asset instance	Physical	GRAI including serial number component = 8003
Individual asset instance	Physical	GIAI including serial number component = 8004

Document instance	Physical or Digital	GDTI including serial number component = 253
Coupon instance	Physical or Digital	GCN including serial number component = 255
Component/part instance	Physical	CPID + AI 8011 (compound) = 8010 + 8011
Unit Pack Unique Identifier (upUI)	Physical (restricted use for compliance with regulations)	= GTIN + AI 235 (compound) = 01 & 235

3.6 Subclass relationship

The structure of a GS1 Web URI is such that:

- identifiers, including identifier qualifiers used to form compound identification keys, are expressed in the URI path information and attributes in the URI query string;
- identifiers are increasingly specific as you move from left to right within the URI path information.

It follows that `https://example.com/gtin/614141123452/cpv/2A` identifies a subclass of `https://example.com/gtin/614141123452`. Semantic applications SHOULD infer this relationship for class-level, i.e. not instance-level items (see section 3.7.4).

3.7 Interpreting the URI query string

The query string in a conformant GS1 Digital Link URI, if present, may contain facts about the identified resource and/or key=value pairs that provide processing instructions for resolvers or for target URLs accessed following redirection from a resolver. The query string does not contain identifiers for the item itself (these are in the path). Therefore, when interpreting the URI to extract facts, the query string SHALL NOT be used as the subject of the derived triples in whole or in part.

Where the key in the query string is a defined GS1 attribute AI, such as an expiry date or measured weight, these SHALL be interpreted as facts about the identified item or class. The GS1 Web vocabulary [WebVoc] provides properties for all GS1 attribute AIs.

3.7.1 Determine if the RDF Subject is a unique instance identifier

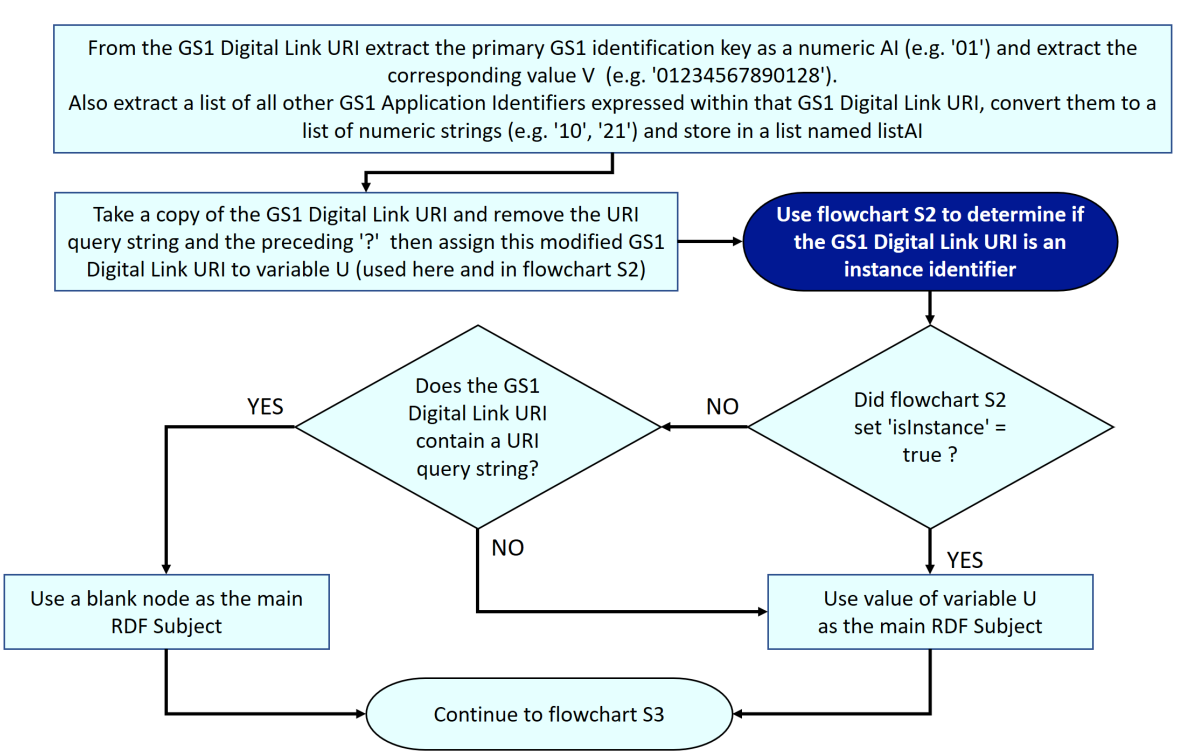
Table S1 provides a list of GS1 Application Identifiers of primary GS1 identification keys that may correspond to unique instance identifiers themselves or as compound keys in combination with additional GS1 Application Identifiers.

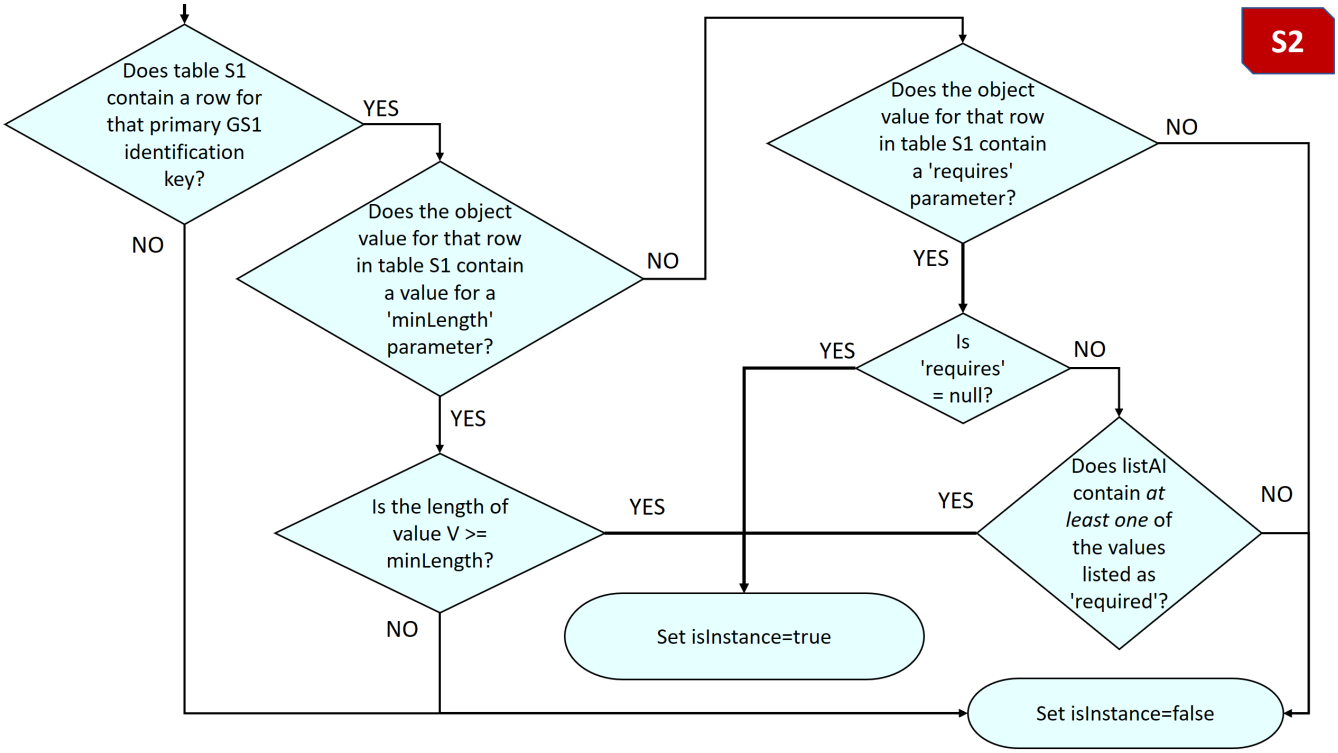
Where the second column, 'requires' is null, the primary identification key already identifies a unique instance. Where the second column shows a list of additional AI values, a unique instance exists provided that one of those additional AIs is specified, together with a value.

The third column indicates a 'mininum length'. For a GS1 primary identification key in table S1, if a minimum length is indicated in the third column and its value is equal or greater than the minimum length, a unique instance exists. This is to support GS1 Application Identifiers that have optional serial components or serial references.

AI of GS1 Primary ID key	Requires additional AIs	Minimum Length
01	21, 235	
00	null	
253		14

AI of GS1 Primary ID key	Requires additional AIs	Minimum Length
254		14
8003		15
8004	null	
8006	21	
8010	8011	



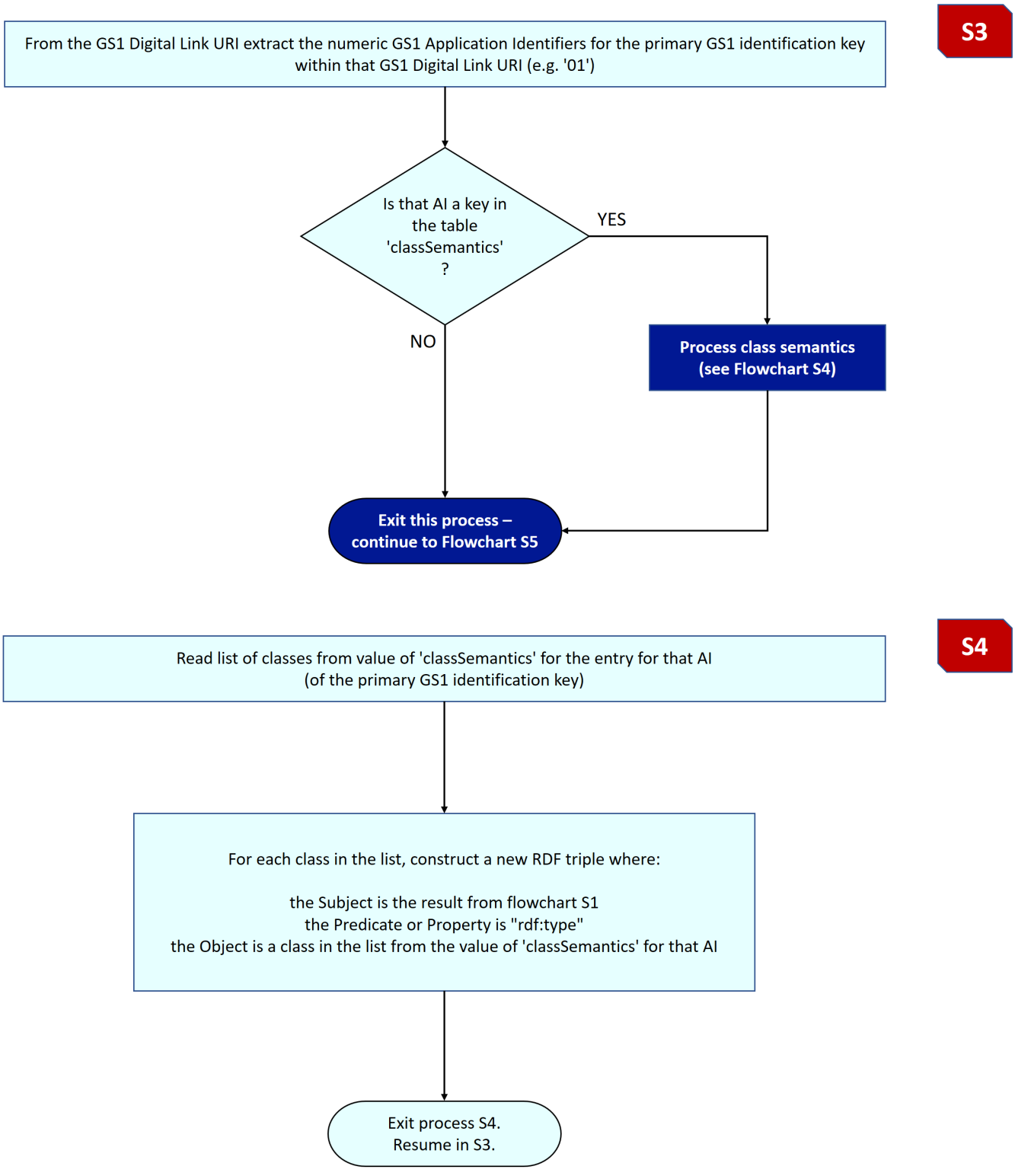


3.7.2 Determine class relationships that can be inferred

Table 'classSemantics' shows relationships between GS1 Application Identifiers of primary GS1 identification keys and corresponding semantic classes that can be inferred for such AIs.

AI of GS1 Primary ID key	Inferred class
01	gs1:Product , schema:Product
8006	gs1:Product , schema:Product
414	gs1:Place , schema:Place
417	gs1:Organization, schema:Organization

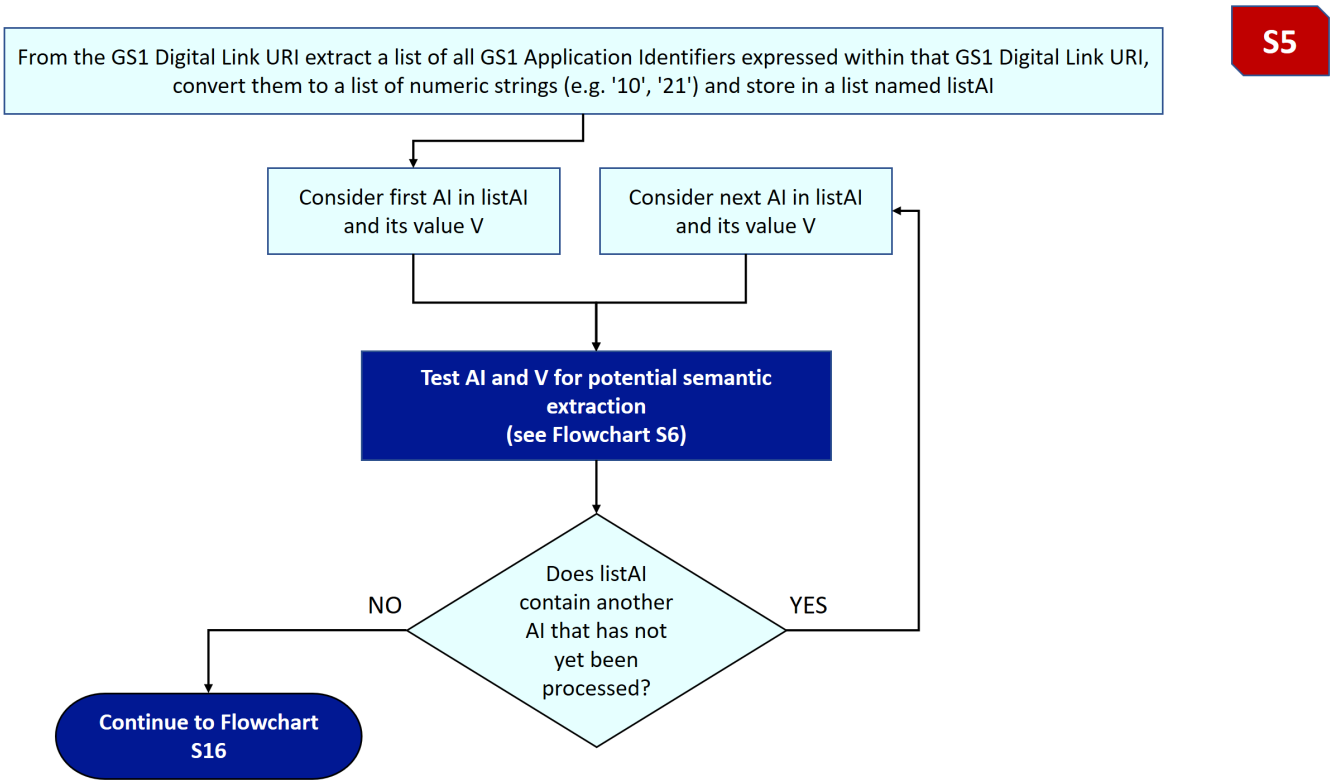
Flowcharts S3 and S4 explain the logic for inferring semantic classes and how to construct the associated RDF triples, with reference to table 'classSemantics'.

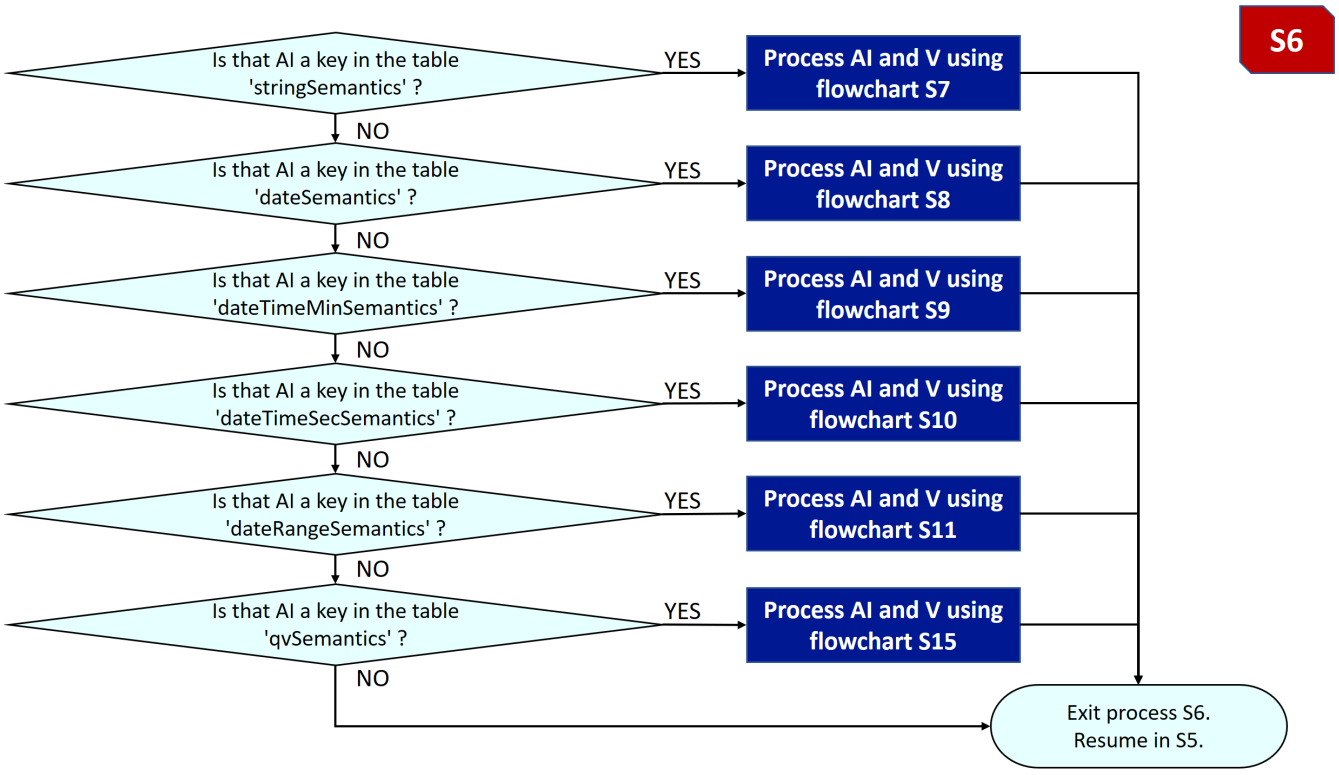


3.7.3 Extract further relationships expecting data values

Many GS1 Application Identifiers express data values about specific identified things. These include data such as net weight, gross weight, dimensions, date of production, expiry date etc. The following subsections explain how to extract further semantic relationships from such AIs, depending on the type and format of their value.

As shown in Flowchart S5, the first step is to convert a GS1 Digital Link URI into a table that maps each GS1 Application Identifier present within the GS1 Digital Link URI to its value, then use Flowchart S6 and its dependent flowcharts (S7 to S15) to perform appropriate reformatting of the value so that it is in a suitable format for use in a Linked Data RDF triple for expressing the semantic relationship.

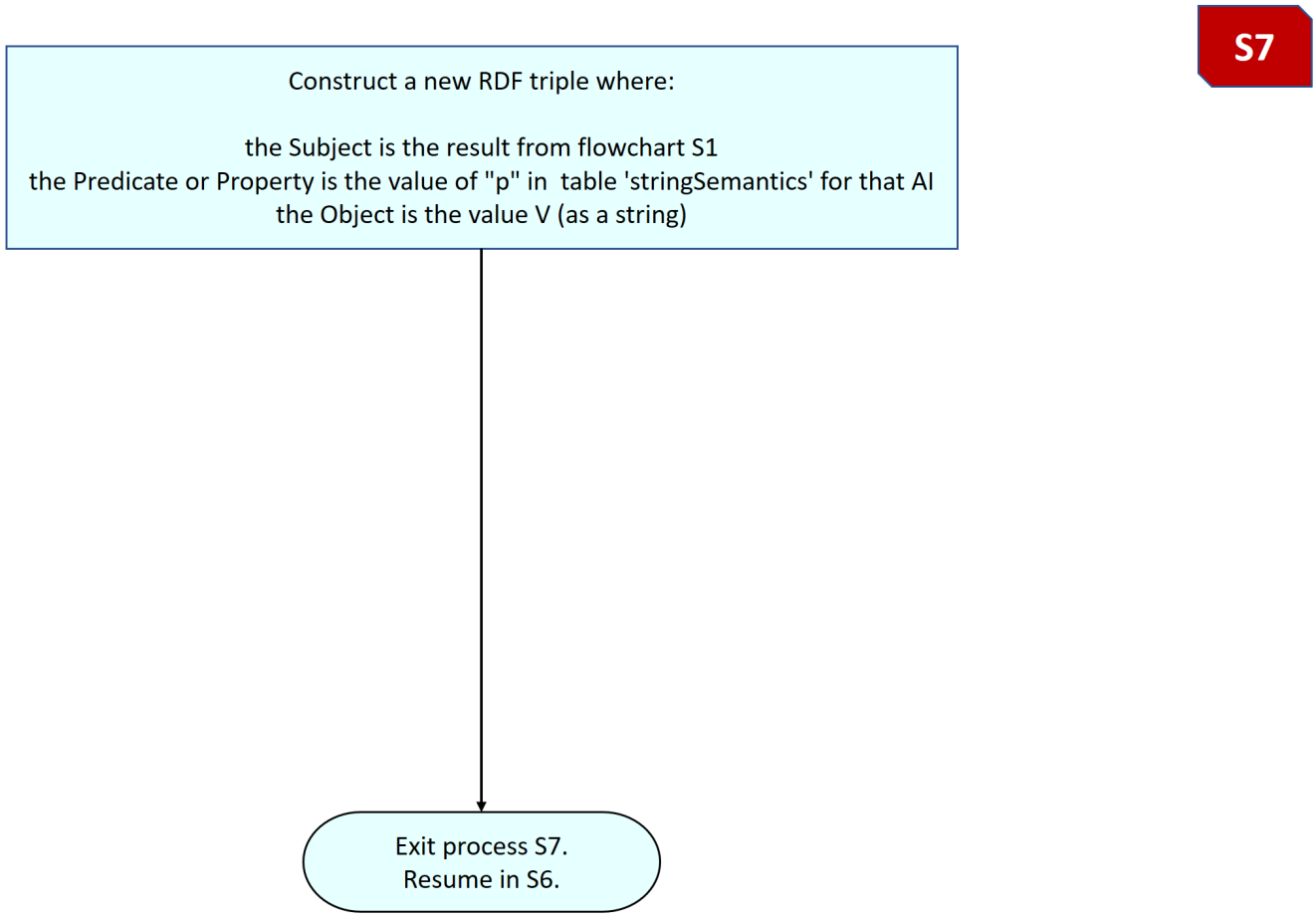




3.7.3.1 Properties expecting string values

Table ‘stringSemantics’ provides mappings between GS1 Application Identifiers that expect a string value and corresponding existing (and future) properties in the GS1 Web vocabulary that also expect an xsd:string value.

GS1 Application Identifier	Corresponding property	Explanation
01	gs1:gtin , schema:gtin	Global Trade Item Number
10	gs1:batchLot	Batch or Lot Number
21	gs1:serialNumber	Serial Number for GTIN (manufacturer-assigned)
22	gs1:cpv	Consumer Product Variant
235	gs1:tpx	Third-party controlled serialised extension to GTIN (assigned by third party)

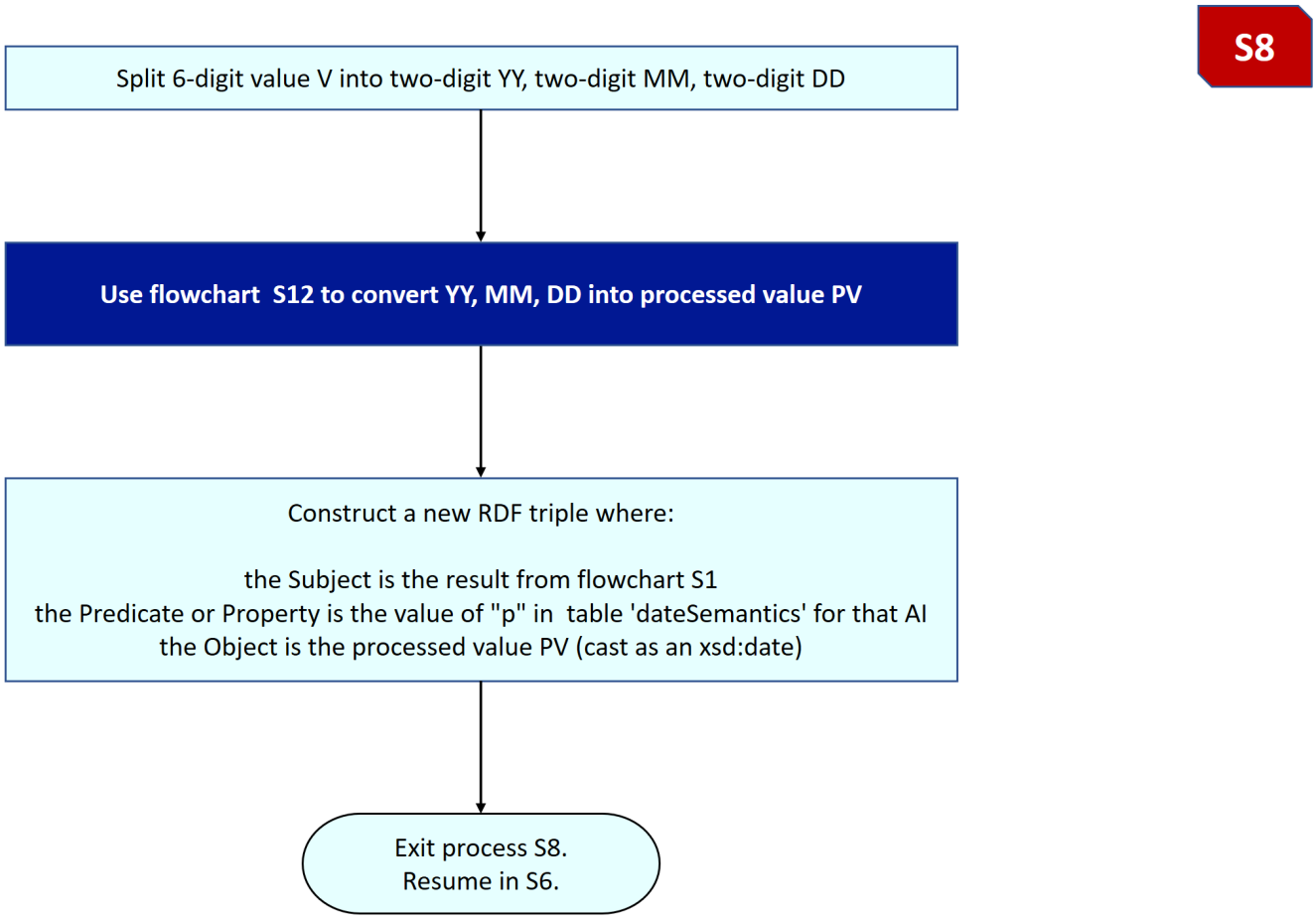


3.7.3.2 Properties expecting date values

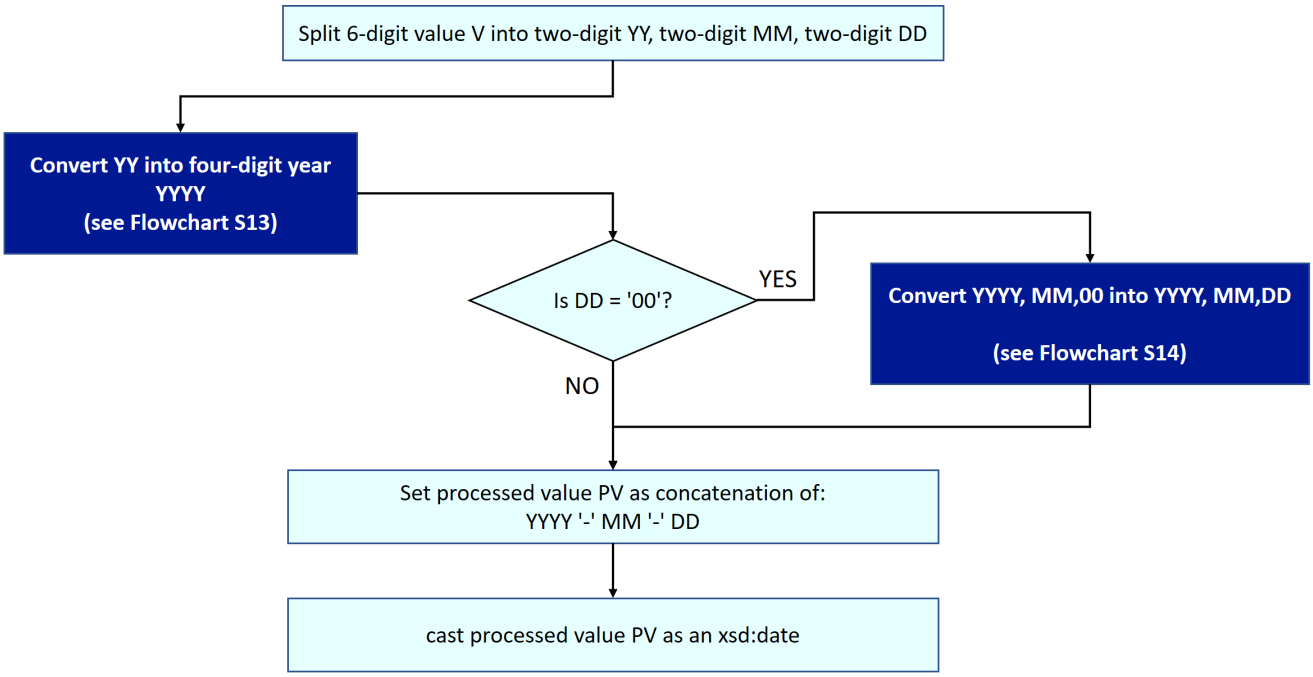
Table ‘dateSemantics’ provides mappings between GS1 Application Identifiers that expect a date value formatted as YYMMDD and corresponding (future) properties in the GS1 Web vocabulary that expect a value cast as an xsd:date.

GS1 Application Identifier	Corresponding property	Explanation
11	gs1:productionDate	Date of Production
12	gs1:dueDate	Due date (for payment of invoice)
13	gs1:packagingDate	Packaging date
15	gs1:bestBeforeDate	Best before date
16	gs1:sellByDate	Sell by date
17	gs1:expirationDate	Expiration date
7006	gs1:firstFreezeDate	First freeze date

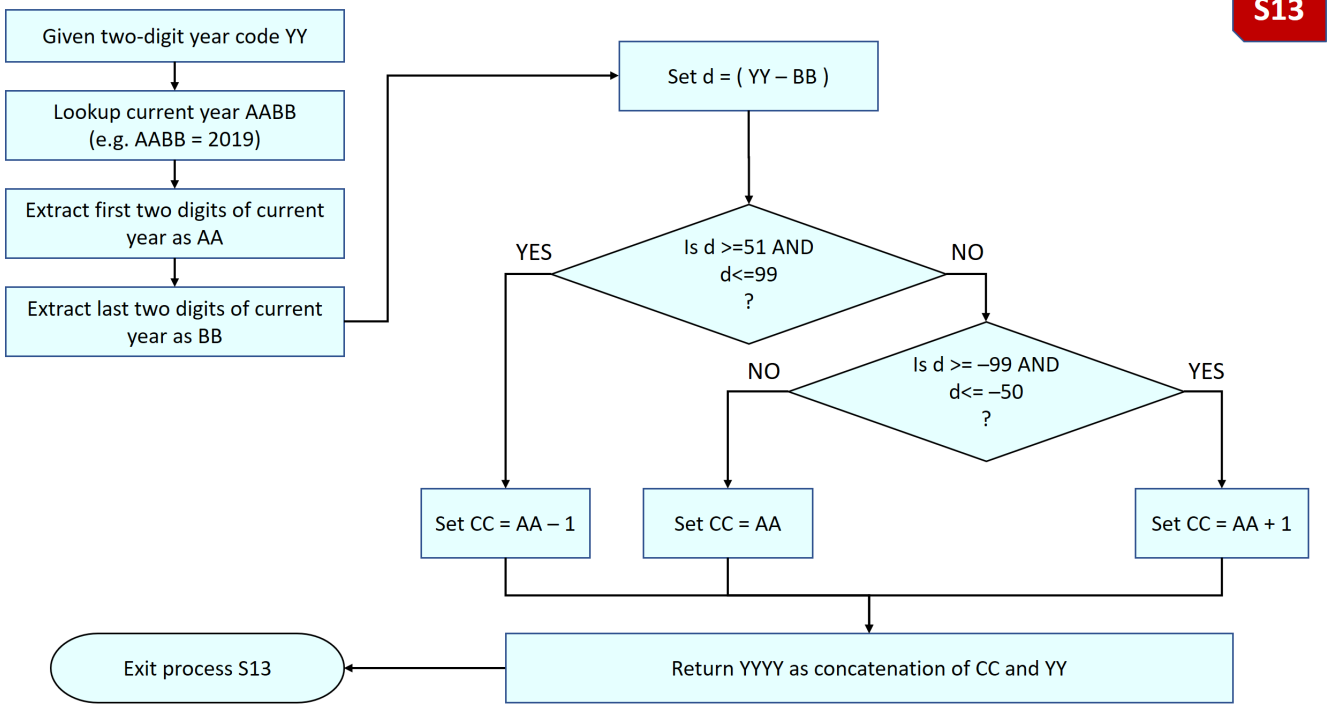
Flowcharts S8, S12, S13 and S14 explains the logic for reformatting a date from format YYMMDD into an xsd:date value. Flowchart S13 handles correct conversion of a two-digit year YY into a four-digit YYYY value. Flowchart S14 handles conversion of a two-digit date expressed as '00' to mean "last day of the month" to the corresponding actual DD, considering the values of MM and YYYY.

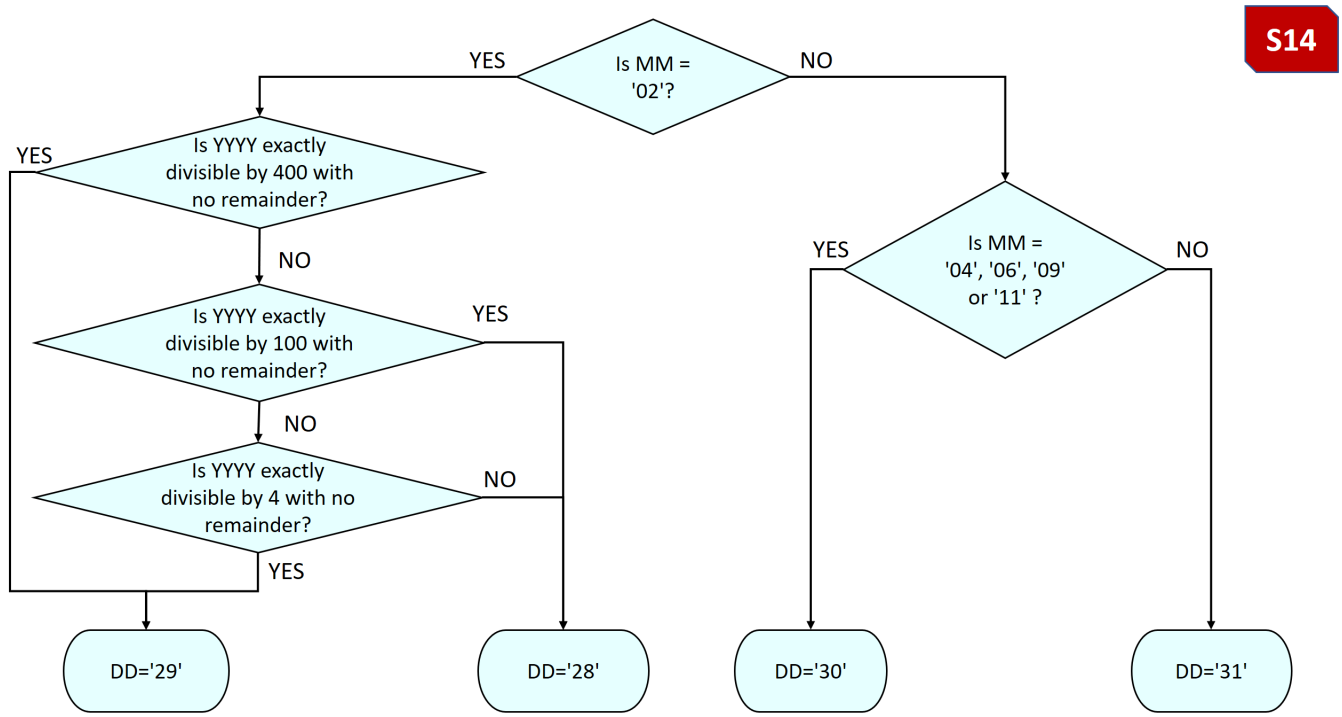


S12



S13

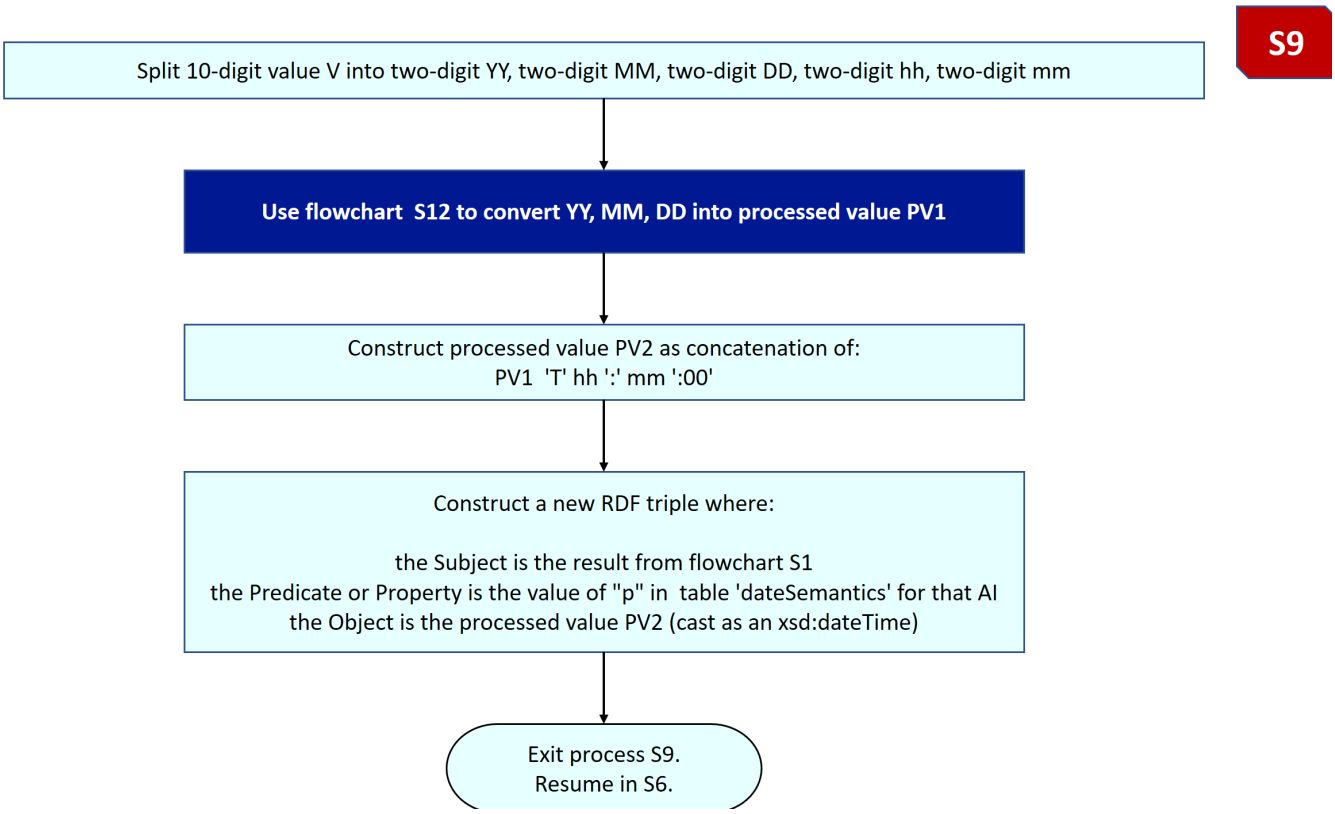




3.7.3.3 Properties expecting dateTime values with minute granularity

Table 'dateTimeMinSemantics' lists the GS1 Application Identifier that expresses date values to granularity in minutes. Flowcharts S9 and S12 explain how to convert that YYMMDDhhmm value into an xsd:dateTime value suitable for use with Linked Data RDF triples.

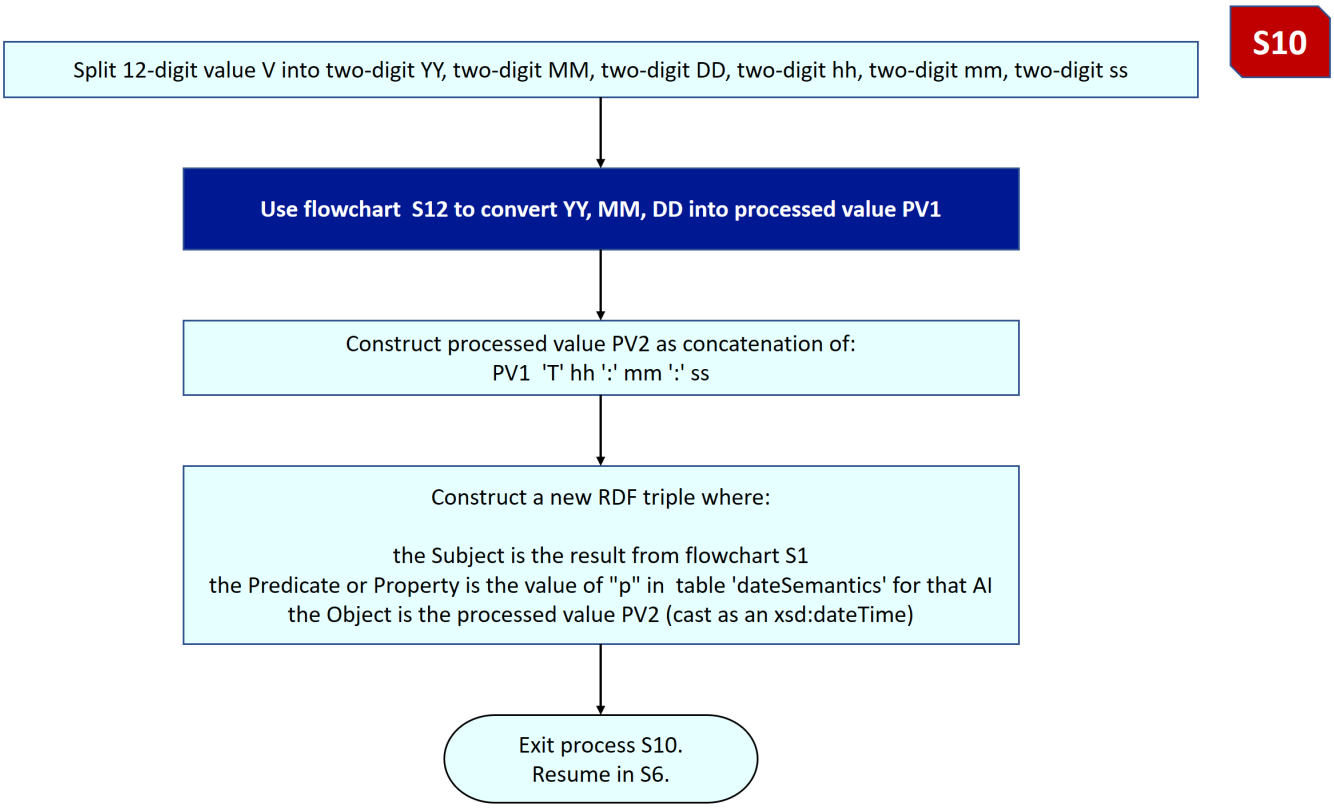
GS1 Application Identifier	Corresponding property	Explanation
7003	gs1:expirationDateTime	Expiration date and time



3.7.3.4 Properties expecting dateTime values with second granularity

Table 'dateTimeSecSemantics' lists the GS1 Application Identifier that expresses date values to granularity in seconds. Flowcharts S10 and S12 explain how to convert that YYMMDDhhmmss value into an xsd:dateTime value suitable for use with Linked Data RDF triples.

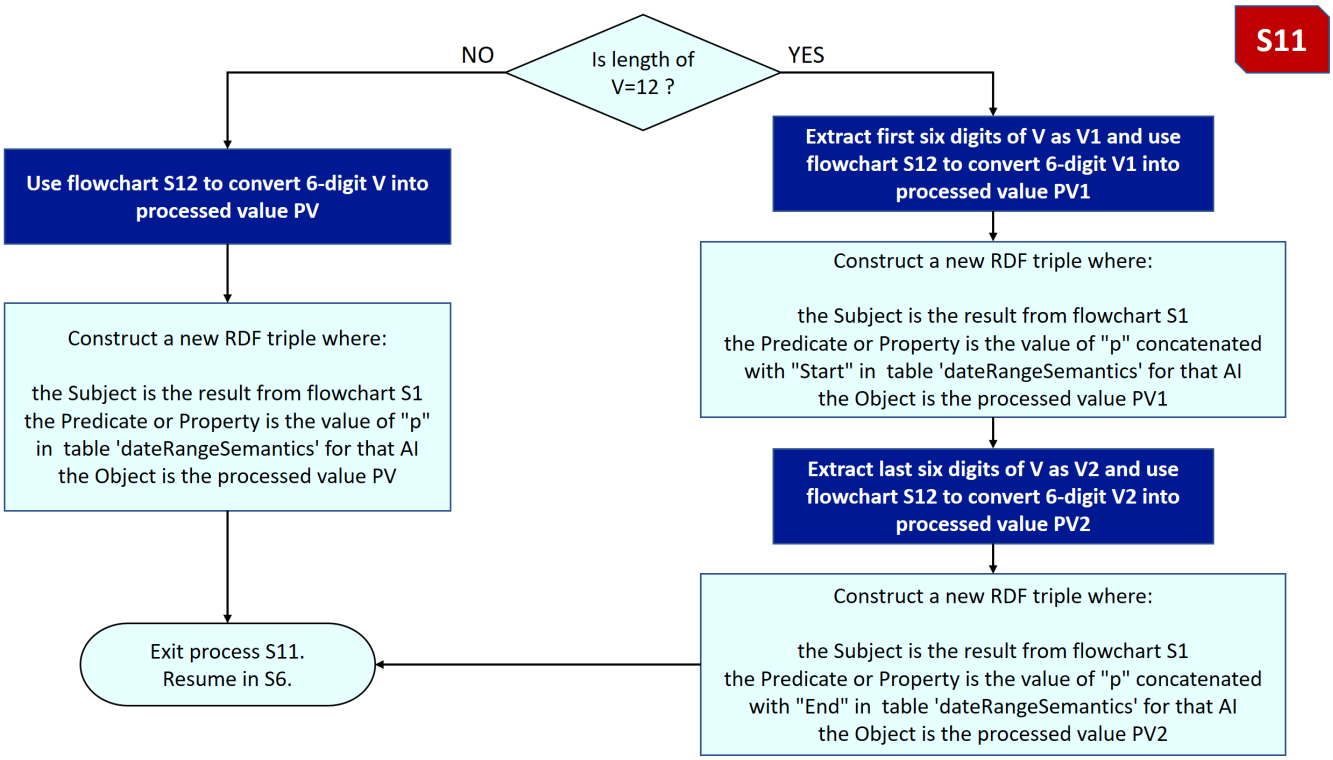
GS1 Application Identifier	Corresponding property	Explanation
8008	gs1:productionDateTime	Date and time of production



3.7.3.5 Properties related to date ranges

Table 'dateRangeSemantics' lists the GS1 Application Identifier that expresses a date or a date range. Flowcharts S11 and S12 explain how to convert that value from format YYMMDD or YYMMDDYYMMDD into an xsd:dateTime value suitable for use with Linked Data RDF triples. Note that where a date range is expressed via a 12-digit value formatted as YYMMDDYYMMDD, additional properties are defined to express the start and end of the date range.

GS1 Application Identifier	Correponding property	Required matching format of value
7007	gs1: harvestDate	YYMMDD
	gs1: harvestDateStart	**YYMMDD**YYMMDD
	gs1: harvestDateEnd	YYMMDD**YYMMDD**



3.7.3.6 Properties expecting quantitative values (value and unit code)

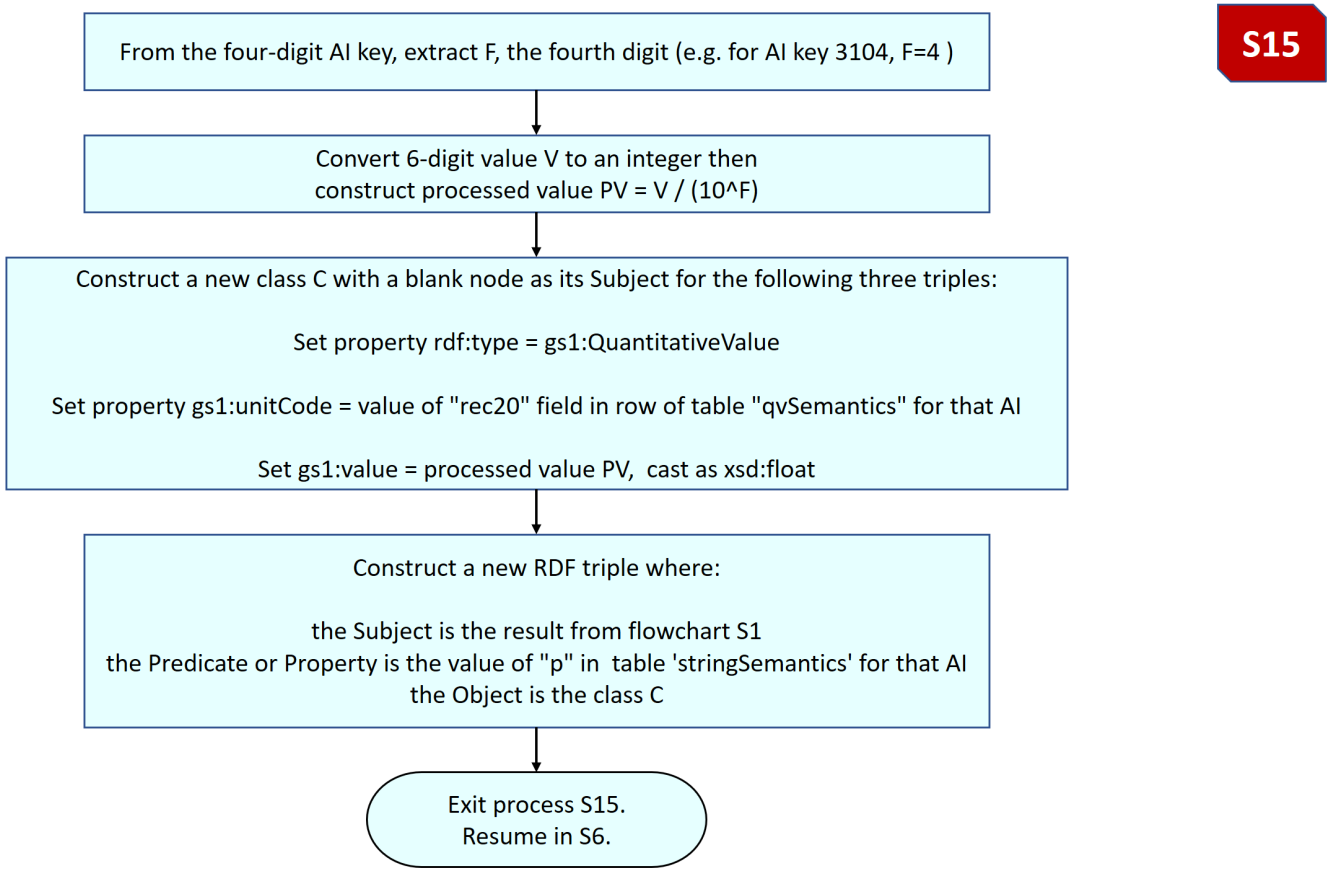
Table 'qvSemantics' below lists GS1 Application Identifiers whose values correspond to quantitative values (gs1:QuantitativeValue, schema:QuantitativeValue) that have a numeric floating-point value and an associated unit code, expressed using the appropriate UN ECE Recommendation 20 unit code. Note that unlike the previous tables, the first column of table 'qvSemantics' expresses a range of GS1 Application Identifiers. For example, considering the first row, 310*n* corresponds to GS1 Application Identifiers 3100, 3101, 3102, 3103, 3104 and 3105 since *n* takes values in the range 0 through 5. The value of the fourth digit (*n*) is meaningful and is used to indicate that the 6-digit value NNNNNN of that GS1 Application Identifier should be divided by 10^n in order to reach the actual value. Note also that several ranges of GS1 Application Identifier correspond to the same semantic property but differ in the unit code associated with each range. For example, 310*n*, 320*n*, 356*n*, 357*n* all correspond to net weight.

Flowchart S15 explains the logic for converting the 6-digit numeric value of the corresponding GS1 Application Identifier into a class (of type gs1:QuantitativeValue or schema:QuantitativeValue) that expresses the numeric value and unit code.

GS1 Application Identifier range (where *n* = 0 through 5)	Corresponding property	UN ECE Recommendation 20 unit code
310*n*	gs1:netWeight	KGM
320*n*	gs1:netWeight	LBR
356*n*	gs1:netWeight	APZ
357*n*	gs1:netWeight	ONZ
330*n*	gs1:grossWeight	KGM
340*n*	gs1:grossWeight	LBR
315*n*	gs1:netContent	LTR
316*n*	gs1:netContent	MTQ

GS1 Application Identifier range (where *n* = 0 through 5)	Corresponding property	UN ECE Recommendation 20 unit code
360*n*	gs1:netContent	QT
361*n*	gs1:netContent	GLL
365*n*	gs1:netContent	FTQ
364*n*	gs1:netContent	INQ
366*n*	gs1:netContent	YDQ
335*n*	gs1:grossVolume	LTR
336*n*	gs1:grossVolume	MTQ
368*n*	gs1:grossVolume	FTQ
367*n*	gs1:grossVolume	INQ
369*n*	gs1:grossVolume	YDQ
363*n*	gs1:grossVolume	GLL
362*n*	gs1:grossVolume	QT
328*n*	gs1:outOfPackageDepth	FOT
327*n*	gs1:outOfPackageDepth	INH
313*n*	gs1:outOfPackageDepth	MTR
329*n*	gs1:outOfPackageDepth	YRD
348*n*	gs1:inPackageDepth	FOT
347*n*	gs1:inPackageDepth	INH
333*n*	gs1:inPackageDepth	MTR
349*n*	gs1:inPackageDepth	YRD
322*n*	gs1:outOfPackageLength	FOT
321*n*	gs1:outOfPackageLength	INH
311*n*	gs1:outOfPackageLength	MTR
323*n*	gs1:outOfPackageLength	YRD
342*n*	gs1:inPackageLength	FOT
341*n*	gs1:inPackageLength	INH
331*n*	gs1:inPackageLength	MTR
343*n*	gs1:inPackageLength	YRD
325*n*	gs1:outOfPackageWidth	FOT
324*n*	gs1:outOfPackageWidth	INH
312*n*	gs1:outOfPackageWidth	MTR
346*n*	gs1:outOfPackageWidth	YRD
345*n*	gs1:inPackageWidth	FOT
344*n*	gs1:inPackageWidth	INH
332*n*	gs1:inPackageWidth	MTR
346*n*	gs1:inPackageWidth	YRD
351*n*	gs1:netArea	FTK

GS1 Application Identifier range (where *n* = 0 through 5)	Corresponding property	UN ECE Recommendation 20 unit code
350*n*	gs1:netArea	INK
314*n*	gs1:netArea	MTK
352*n*	gs1:netArea	YDK
354*n*	gs1:grossArea	FTK
353*n*	gs1:grossArea	INK
334*n*	gs1:grossArea	MTK
355*n*	gs1:grossArea	YDK
337*n*	gs1:massPerUnitArea	28

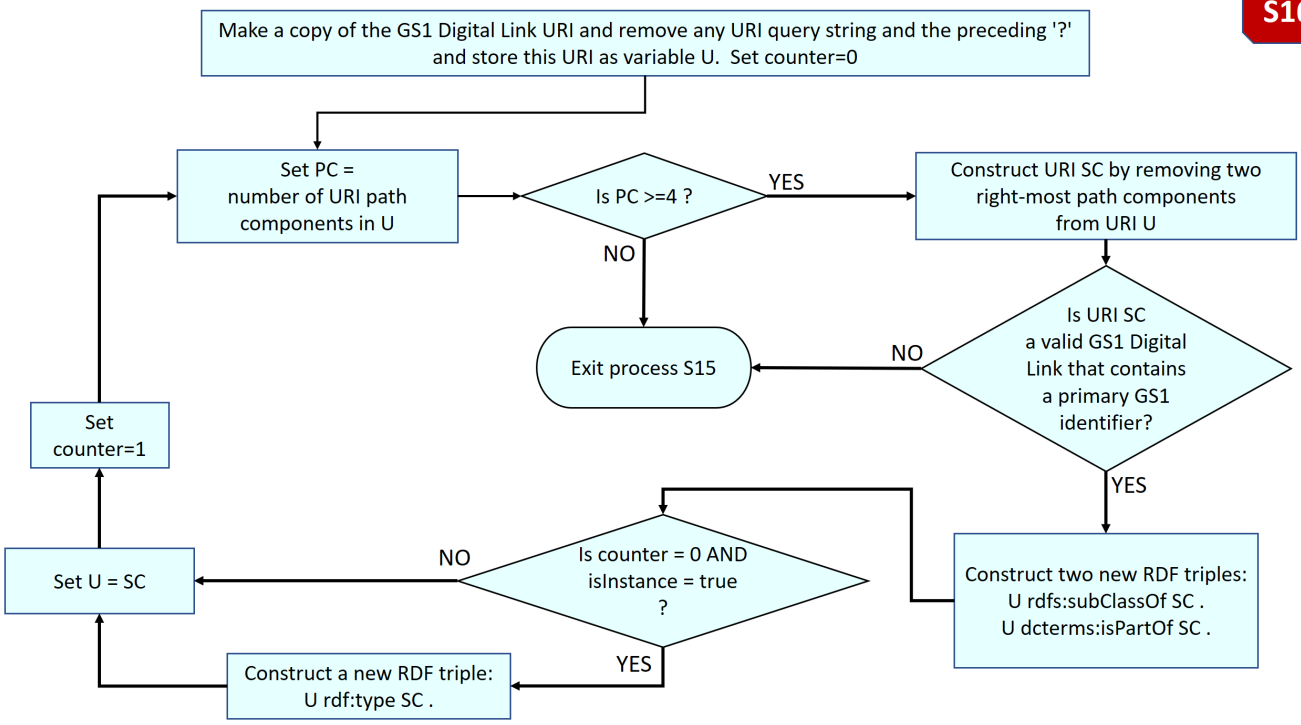


3.7.4 Extract subclass relationships

The URI path information of a GS1 Digital Link URI contains hierarchical information such that successive path components result in increasingly finer levels of granularity of identification. Semantically each successive pair of path components beyond the pair that corresponds to the primary GS1 identification key expresses a subclass or a subclass of a subclass, etc.

Flowchart S16 explains the logic for extraction of such subclass relationships from a GS1 Digital Link URI.

S16



3.8 Worked examples

This subsection is informative

To explain the semantics in more detail we'll work through some examples.

3.8.1 GTIN + CPV

This Digital Link URI includes a GTIN and a consumer product variant

`https://example.com/gtin/614141123452/cpv/2A`

and should be interpreted as follows:

```
<https://example.com/gtin/614141123452/cpv/2A> a schema:Product, gs1:Product;
  rdfs:subClassOf <https://example.com/gtin/614141123452>;
  gs1:gtin "614141123452";
  schema:gtin "614141123452";
  gs1:cpv "2A".
```

Because the primary identifier is a GTIN, this is a `schema:Product` and a gs1:Product

We know that this is a consumer product variant of the given GTIN and this is expressed as a sub class relationship.

The GTIN and CPV are both given directly in the Digital Link URI but there is no query string containing attributes about a product instance, therefore we can make assertions at the class level using identifiers in the URI.

3.8.2 GTIN + batch/lot + Serial number + expiry date

This Digital Link URI includes a GTIN and a serial number

`https://example.com/gtin/614141123452/lot/ABC/ser/00001?exp=190400`

Importantly, this is an instance level identifier and so we can assign the attributes in the query string to the identified product instance to give the interpretation as follows:

```
<https://example.com/gtin/614141123452/lot/ABC/ser/00001> a schema:Product,
gs1:Product;
  rdfs:subClassOf <https://example.com/gtin/614141123452/lot/ABC>;
  dcterms:isPartOf <https://example.com/gtin/614141123452/lot/ABC>;
  gs1:serialNumber "00001";
  gs1:expirationDate "2019-04-30"^^xsd:date;
  skos:notation "(01)00614141123452(10)ABC(21)00001"^^gs1:ElementeStrings.
```

We can further assert:

```
<https://example.com/gtin/614141123452/lot/ABC>
  rdfs:subClassOf <https://example.com/gtin/614141123452>;
  gs1:gtin "614141123452";
  schema:gtin "614141123452";
  gs1:batchLot "ABC”;
  skos:notation "(01)00614141123452(10)ABC"^^gs1:ElementeStrings.
```

3.8.3 GTIN + Measured Weight

This GS1 Digital Link URI includes a GTIN and a measured weight

```
https://example.com/gtin/6141411234523103=000500
```

In this case, we have a query string that provides attributes of the item, its measured weight, and so we have to use a blank node to represent the product instance rather than make the obviously false assertion that all instances of this product have the same measured weight. The semantic interpretation is therefore as follows:

```
[] a <https://example.com/gtin/614141123452>, schema:Product, gs1:Product;
  gs1:gtin "614141123452";
  schema:gtin "614141123452";
  gs1:netWeight [a gs1:QuantitativeValue; gs1:unitCode "KGM"; gs1:value "0.5"];
  skos:notation "(01)00614141123452(3103)000500"^^gs1:ElementStrings.
```

3.9 AIs defined as Web vocabulary terms

To support the semantic interpretation of GS1 Digital Link URIs, we propose that the following terms be added to the GS1 Web vocabulary. These include additional properties to express GS1 Application Identifiers that serve as qualifiers of primary GS1 identification keys. These include:

Proposed new property	GS1 Application Identifier	Range of new property
gs1:cpv	22	xsd:string
gs1:batchLot	10	xsd:string
gs1:serialNumber	21	xsd:string
gs1:tpx	235	xsd:string

They also include properties that correspond to GS1 Application Identifiers that express various date values and dateTime values. These include:

Proposed new property	GS1 Application Identifier	Range of new property	Format of AI value
gs1:productionDate	11	xsd:date	YYMMDD
gs1:packagingDate	13	xsd:date	YYMMDD
gs1:bestBeforeDate	15	xsd:date	YYMMDD
gs1:sellByDate	16	xsd:date	YYMMDD
gs1:expirationDate	17	xsd:date	YYMMDD
gs1:firstFreezeDate	7006	xsd:date	YYMMDD
gs1:dueDate	12	xsd:date	YYMMDD
gs1:harvestDate	7007	xsd:date	YYMMDD
gs1:harvestDateStart	7007	xsd:date	**YYMMDD**YYMMDD
gs1:harvestDateEnd	7007	xsd:date	YYMMDD**YYMMDD**
gs1:productionDateTime	8008	xsd:dateTime	YYMMDDhh[mm][ss]
gs1:expirationDateTime	7003	xsd:dateTime	YYMMDDhhmm

Additionally, although the GS1 Web vocabulary already defines many properties relating to net weight, gross weight, net content and various dimensions of a product, a number of additional properties need to be defined:

Proposed new property	GS1 Application Identifier range (n=0 to 5)	Range of new property
gs1:grossVolume	335n 336n 362n 363n 367n 368n 369n	gs1:QuantitativeValue
gs1:netArea	314n 350n 351n	gs1:QuantitativeValue
gs1:grossArea	334n 353n 354n 355n	gs1:QuantitativeValue
gs1:massPerUnitArea	337n	gs1:QuantitativeValue

3.10 Link type semantics

The majority of link types are defined in the GS1 namespace (https://ref.org/voc/) but the Web vocabulary includes some assertions made about useful terms that already exist in schema.org. In terms of semantics, we define a super property of `gs1:linkType` that has a domain of `owl:Thing` and a range of `xsd:anyURI`, i.e. it is very generic. Each GS1 link type is a sub property of this and typically has further restrictions. We assert that a small number of schema.org properties are also sub properties of `gs1:linkType`. This has no effect on their semantics but does allow tools to present a list of link types easily.

Where possible, the Web vocabulary includes matches to schema.org properties using the relevant SKOS vocabulary term [SKOS]. Resolvers MAY use this information to include schema.org as additional link types when exposing available links. For example, this formal relationship is asserted:

```
gs1:pip skos:broader schema:url
```

This means that when exposing the link to the product information page, the `@rel` value on that link MAY be "`gs1:pip schema:url`" since both relations apply.

4 Changes since version 1.1.3

Where versions 1.1.1 through 1.1.3 all retained the original content but warned that it has been superseded, this version removes all that content entirely. Note that all previous versions are available as immutable documents through ref.gs1.org.

This document attempts to clarify the remaining content and provide context for its possible use. Its primary function is as an archive and reference point for future work, the Compression Technical Standard for EPC binary strings being the primary example.

5 References

[Arch]

GS1 System Architecture, see https://ref.gs1.org/architecture/system-architecture/

[BCP 190]

URI Design and Ownership. M. Nottingham, IETF June 2020 https://www.rfc-editor.org/rfc/rfc8820.html

[DL-11]

GS1 Digital Link version 1.1.0 Mark Harrison, Phil Archer, Dominique Guinard, Laurent Tonnelier, Grant Courtney, Tony Rodrigue, Marie Petre, Rigo Wenning. GS1 ratified standard, February 2020 https://ref.gs1.org/standards/digital-link/1.1.0/

[DL-C]

GS1 Digital Link URI: Compression Technical Standard for EPC binary strings. Mark Harrison. Ratified GS1 standard, July 2025 https://ref.gs1.org/standards/digital-link/compression/

[DL-URI]

GS1 Digital Link Standard: URI Syntax. Mark Harrison, Peta Ding, Phil Archer. GS1 Ratified standard, January 2021 and revisions. https://ref.gs1.org/standards/digital-link/uri-syntax/

[EST]

GS1 Retail 2D barcode size estimator. Mark Harrison. https://ref.gs1.org/tools/module-count

[GENSPECS]

GS1 General Specifications. Ratified GS1 standard, updated and ratified annually https://ref.gs1.org/standards/genspecs/

[LMS]

GS1 Lightweight Messaging Standard for Verification of Product Identifiers. Craig Alan Repec, Mark Harrison, John Ryu. GS1 ratified standard July 2019 https://www.gs1.org/sites/default/files/docs/standards/gs1_lightweight_verification_messaging_standard_v1-1.pdf

[Resolver]

GS1-Conformant resolver standard. Phil Archer. GS1 ratified standard February 2025. https://ref.gs1.org/standards/resolver/

[RFC 2119]

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF RFC March 1997 https://www.rfc-editor.org/rfc/rfc2119.html

[RFC 4648]

The Base16, Base32, and Base64 Data Encodings. S. Josefsson. IETF RFC October 2006 https://www.rfc-editor.org/rfc/rfc4648.html

[SKOS]

SKOS Simple Knowledge Organization System. Alastair Miles, Sean Bechhofer. W3C Recommendation 8 August 2009 https://www.w3.org/TR/skos-reference/

[TDS]

GS1 EPC Tag Data Standard. Craig Alan Repec, Mark Harrison. GS1 ratified standard February 2025 https://ref.gs1.org/standards/tds/

[TDT]

GS1 EPC Tag Data Translation Standard. Craig Alan Repec, Mark Harrison, Nick Porter. GS1 ratified standard February 2025 https://ref.gs1.org/standards/tdt/

[WebArch]

Architecture of the World Wide Web, Volume One. Ian Jacobs, Norman Walsh. W3C Recommendation 15 December 2004. https://www.w3.org/TR/webarch/

[WebVoc]

The GS1 Web Vocabulary https://ref.gs1.org/voc/

A Intellectual Property

A.1 Introduction and Disclaimer

GS1 seeks to minimise barriers to the adoption of its standards and guidelines by making the intellectual property required to implement them available, to the greatest extent possible, on a royalty-free basis, or when necessary, under a RAND licence. Such royalty-free and RAND licences are provided pursuant to the GS1 IP Policy (available here: https://www.gs1.org/standards/ip), which governs the work of work group participants who contribute to drafting standards and guidelines, including the GS1 Digital Link Standard Release 1.1 (for the purpose of this paragraph A.1, the "**standard**"). In addition to licences, the GS1 IP Policy provides various benefits and obligations that apply to all implementers of GS1 standards and guidelines, and all implementations of GS1 standards are subject to those terms.

Nevertheless, please note the possibility that an implementation of one or more features of this standard or guideline may be the subject of a patent or other intellectual property right that is not covered by the licences granted pursuant to the IP Policy. In addition, the licences granted under the IP Policy do not include the IP rights or claims of third parties who were not participants in the corresponding standard development work group.

Accordingly, GS1 recommends that any person or organisation developing an implementation of this standard or guideline should determine whether any patents or other intellectual property may encompass such implementation, and whether a licence under a patent or other IP right is needed. The implementer should determine the potential need for licensing in view of the details of the specific implementation being designed in consultation with that party's patent counsel.

The official versions of all GS1 standards and guidelines are provided as PDF files on GS1's online reference directory (https://ref.gs1.org) (the "GS1 Reference"). Any other representations of standards or guidelines in any other format (e.g., web pages) are provided for convenience and descriptive purposes only, and in the event of a conflict, the GS1 Reference document shall govern.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR PARTICULAR PURPOSE, ACCURACY OR COMPLETENESS, OR ANY WARRANTY OTHERWISE ARISING OUT OF THIS DOCUMENT. GS1 disclaims all liability for any damages arising from any 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 makes no commitment to update the information contained herein, and retains the right to make changes to this document at any time, without notice. GS1® and the GS1 logo are registered trademarks of GS1 AISBL.

A.2 Notices

A.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.2.1, the "**Patent and patent applications**") who participated to the Work Group designing the GS1 Digital Link Standard Release 1.1, gave notice that they believe the Patent and patent applications contain Essential Claims for the implementation of the GS1 Digital Link Standard Release 1.1 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 1.1:

- 2.5 Structure of the compressed string
- 2.8 Formal ABNF grammar for compressed GS1 Digital Link URIs
- 2.9 Compression procedure and flowcharts
- 2.10 Decompression procedure and flowcharts

For the patent and if the patent applications materialise, Mobilead 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 Standard Release 1.1. Implementations that extend in implementation what is described in GS1 Digital Link Standard Release 1.1 can contact Mobilead SAS for licensing.

The statements given in this section are from the patent holders. GS1 makes no explicit or implicit representation on the validity of the aforementioned patent nor on the claimed presence of Essential Claims

reading on the GS1 Digital Link Standard Release 1.1. This note is not legal advice and GS1 takes no responsibility for the accuracy, legal correctness or fitness for any purpose of the information provided in this section.

A.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.2.2, the "**Patent Application**"), who participated to the Work Group designing the GS1 Digital Link Standard Release 1.1, gave notice that they believe their Patent Application contains Essential Claims for the implementation of the GS1 Digital Link Standard Release 1.1 for the following sections:

- 2. 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 Standard Release 1.1. Implementations that extend in implementation what is described in GS1 Digital Link Standard Release 1.1 can contact Servicetag SAS for licensing.