



The Global Language of Business

*Please note that chapters 4-6 of this standard have been superseded by GS1 Digital Link Standard: URI Syntax available from <https://ref.gs1.org/standards/digital-link/uri-syntax/>.
Section 7 has been superseded by GS1-Conformant resolver standard (<https://ref.gs1.org/standards/resolver/>)
All other sections are the current version of the standard.*

GS1 Digital Link Standard

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

Release 1.1.3, Ratified, Mar 2024

Document Summary

Document Item	Current Value
Document Name	GS1 Digital Link Standard
Document Date	Mar 2024
Document Version	1.1
Document Issue	3
Document Status	Ratified
Document Description	enabling consistent representation of GS1 identification keys within web addresses to link to online information and services

Contributors

Name	Organisation
Grant Courtney (Chair)	GlaxoSmithKline
Dominique Guinard (Chair)	EVERYTHING
Laurent Tonnelier (Chair)	SERVICETAG/MOBILEAD
Stefan Artlich	Bayer AG – Division Pharma
Attilio Bellman	Antares Vision
Adam Björnstjerna	HKScan Sweden AB
Wes Bloemker	Arthrex Inc.
Stephen Brown	Mead Westvaco
Greg Buckley	PepsiCo, Inc.
Henk Dannenberg	NXP Semiconductors
Jeanne Duckett	Avery Dennison RFID
Vera Feuerstein	Nestlé
Tarik Gemei	Amgen Inc.
Viktoria Grahnen	AbbVie
Mike Kuhno	Avery Dennison RFID
Sébastien Langlois-Berthelot	F. Hoffmann-La Roche Ltd.
Nicolas Lecocq	L'Oreal
Phill Marley	AstraZeneca Pharmaceuticals
Yi Meng	Qingdao Haier Washing Machine Co. Ltd.
Martin Neselius	BillerudKorsnäs (Venture)
Tatjana Pathare	F. Hoffmann-La Roche Ltd.
Guilhem Rousselet	SANOFI
Sophie-Mareen Scholz	Dr. August Oetker Nahrungsmittel KG
Mandeep Sodhi	Nestlé
John Terwilliger	Abbott
Gina Tomassi	PepsiCo, Inc.
Julie Vargas	Avery Dennison RFID
Sylvie Vilcoq	DANONE PRODUITS FRAIS FRANCE

Name	Organisation
Evan Bacchus	Costco Wholesale
Fabien Calleia	SIZEASE
Yolande Diaz	Carrefour
Sven Dienelt	Hermann Hagemeyer GmbH & Co. KG
Marcel Ducceschi	Migros-Genossenschafts-Bund
Jonas Elander	Axfood Sverige AB
Max Engström	H&M
Sylvia Rubio Alegren	ICA Sverige AB
Hans Peter Scheidt	C & A SCS
Joachim Wilkens	C & A SCS
Hajo Reissmann	Universitaetsklinikum Schleswig-Holstein
Marc Damhösl	Schweizerische Bundesbahnen SBB
Hirokazu Nagai	Japan Pallet Rental Corporation
Vladimir Dzalbo	Smartrac Technology Germany GmbH
Richard Fisher	DoD Logistics AIT Standards Office
Steven Robba	1WorldSync, Inc.
Karen Arkesteyn	GS1 Belgium & Luxembourg
Koji Asano	GS1 Japan
Andrea Ausili	GS1 Italy
Xavier Barras	GS1 France
Jonas Batt	GS1 Switzerland
Rahma Benfraj	GS1 France
Mats Bjorkqvist	GS1 Sweden
Arnaud Bonnefoy	GS1 France
Jonas Buskenfried	GS1 Sweden
Emanuela Casalini	GS1 Italy
Madalina Cernat	GS1 Romania
Anthony Chan	GS1 Hong Kong
Shawn Chen	GS1 Thailand
Luiz Costa	GS1 Brasil
Benjamin Couty	GS1 France
Tim Daly	GS1 Ireland
Owen Dance	GS1 New Zealand
Michael Davis	GS1 Australia
Kevin Dean	GS1 Canada
Raymond Delnicki	GS1 US
Huipeng Deng	GS1 China
Sean Dennison	GS1 Ireland
Peta Ding	GS1 UK
Ferran Domenech Fuste	GS1 Spain
Xiaowen Dong	GS1 China

Name	Organisation
Guilherme França	GS1 Brasil
Michele Francis Padayachee	GS1 South Africa
Jesper Kervin Franke	GS1 Denmark
Jean-Christophe Gilbert	GS1 France
Vanessa Giulieri	GS1 Italy
Esther Goh	GS1 Singapore
Heinz Graf	GS1 Switzerland
magali Granger	GS1 France
János Gyuris	GS1 Hungary
Rami Habbal	GS1 UAE
Jason Hale	GS1 UK
Karolin Harsanji	GS1 Sweden
Gary Hartley	GS1 New Zealand
Bernie Hogan	GS1 US
Fredrik Holmström	GS1 Sweden
Hideki Ichihara	GS1 Japan
Yoshihiko Iwasaki	GS1 Japan
Yo Han Jeon	GS1 Korea
Yohan Jeon	GS1 Korea
Fiona Jia	GS1 China
Ricky Jones	GS1 UK
Sang Ik Jung	GS1 Korea
Kimmo Keravuori	GS1 Finland
Kazuna Kimura	GS1 Japan
Alexey Krotkov	GS1 Russia
Chris Lai	GS1 Hong Kong
Ildikó Lieber	GS1 Hungary
Marisa Lu	GS1 Chinese Taipei
Ilka Machermer	GS1 Germany
Noriyuki Mama	GS1 Japan
Roberto Matsubayashi	GS1 Brasil
Riad Mechtari	GS1 Algeria
Jan Merckx	GS1 Belgium & Luxembourg
Ephraim Mokheseng	GS1 South Africa
Adrien Molines	GS1 France
Naoko Mori	GS1 Japan
Daniel Mueller-Sauter	GS1 Switzerland
Prince Namane	GS1 South Africa
Daisuke Negishi	GS1 Japan
Eric Ng	GS1 Hong Kong
Staffan Olsson	GS1 Sweden

Name	Organisation
Michel Ottiker	GS1 Switzerland
Manos Papadakis	GS1 Association Greece
Sebastián Perazzo	GS1 Argentina
Bijoy Peter	GS1 India
Sarina Pielaat	GS1 Netherlands
Neil Piper	GS1 UK
Emma Potter	GS1 Australia
Paul Reid	GS1 UK
Alexandre Rieucou	GS1 France
Zbigniew Rusinek	GS1 Poland
Nick Rusman	GS1 Netherlands
Roxana Saravia Bulmini	GS1 Argentina
Sue Schmid	GS1 Australia
Eugen Sehorz	GS1 Austria
Xiaojing Shao	GS1 China
Yuko Shimizu	GS1 Japan
Marcel Sieira	GS1 Australia
Sean Sloan	GS1 Australia
Olga Soboleva	GS1 Russia
Andrew Steele	GS1 Australia
Sylvia Stein	GS1 Netherlands
Jo Anna Stewart	GS1 US
Flora Sue	GS1 China
Ralph Troeger	GS1 Germany
Vivian Underwood	GS1 US
Frits van den Bos	GS1 Netherlands
Ricardo Verza Amaral Melo	GS1 Brasil
Linda Vezzani	GS1 Italy
Rocio Vizcarra	GS1 Argentina
Amber Walls	GS1 US
Yi Wang	GS1 China
Shu Wang	GS1 China
Achim Wetter	GS1 Germany
Stephan Wijnker	GS1 Australia
Ruoyun Yan	GS1 China
Victor Zhang	GS1 China
Marc Blanchet	Viagenie
Adnan Alattar	Digimarc
Philip Allgaier	bpcompass GmbH
Shreenidhi Bharadwaj	Syndigo
Dalibor Biscevic	Business Technologies Ltd

Name	Organisation
Randy Burd	MultiAd Kwiikee
Shawn Cady	Syndigo
Tony Ceder	Charmingtrim
Robert Celeste	Center for Supply Chain Studies
Patrick Chanez	INEXTO SA
Dilip Daswani	Qliktag Software (formally Zeebric LLC)
Cory Davis	Digimarc
Christophe Devins	Adents High-Tech International
Roland Donzelle	SQUARE / TINTAMAR
Chuck Evanhoe	Evanhoe & Associates, Inc.
Susan Flake	Zebra Technologies Corporation
Tomaz Frelj	Četrta pot,d.o.o.,Kranj
Mathieu Gallant	Optel Group
Ivan Gonzalez	recycl3R
Richard Graves	Phy
Danny Haak	Nedap
Mark Harrison	Milecastle Media Limited
John Herzig	Barcode Graphics Inc Canada
Dan James	Digimarc
Paul Kanwar	ScanTrust
J.D. Kern	Syndigo
Thomas Kühne	Goodstag GmbH
André Machado	TrustaTAG
Joel Meyer	Digimarc
Attila Sándor Nagy	infiCom.EU Co. Ltd.
ilteris oney	ecomis
Tiphaine Paulhiac	Ambrosus Technologies
Justin Picard	ScanTrust
Tony Rodriguez	Digimarc
Zbigniew Sagan	Advanced Track and Trace
Joannie Sauvageau	Optel Group
Kim Simonalle	Qliktag Software (formally Zeebric LLC)
Andrew Verb	Bar Code Graphics, Inc.
Elizabeth Waldorf	TraceLink
Alex Winiarski	Winiarski Group
George Wright IV	Product Identification & Processing Systems
Tony Zhang	Syndigo
Pete Alvarez	GS1 Global Office
Phil Archer	GS1 Global Office
Henri Barthel	GS1 Global Office
Robert Beideman	GS1 Global Office

Name	Organisation
Steven Keddie	GS1 Global Office
Jeanette McVeigh	GS1 Global Office
Aaron Miller	GS1 Global Office
Dan Mullen	GS1 Global Office
Craig Alan Repec	GS1 Global Office
Greg Rowe	GS1 Global Office
Kevin Stark	GS1 Global Office

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 <i>GS1 Web URI Structure Standard</i>
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	<p>Addition of chapter on compression/decompression, Addition of semantics chapter</p> <p>Evolutionary steps as decided by the working group concerning, for example, link type being the default, cf. 'a link', introduction of concept of 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.</p> <p>Addition of AIs 417, 235 and 7040</p>
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 Macheimer, 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 10
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

Disclaimer

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

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

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

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

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

Table of Contents

- 1 Executive Summary 12**
- 2 Introduction 13**
- 3 Definitions and namespaces 13**
 - 3.1 Typographical conventions used in this document 14
 - 3.2 Conformance to this standard 14
- 4 What is a URI? 15**
 - 4.1 The GS1 Digital Link URI 18
- 5 GS1 Digital Link URI Syntax 19**
 - 5.1 Character sets 19
 - 5.2 Primary identification keys 20
 - 5.3 Key qualifiers 21
 - 5.4 Primary key formats 22
 - 5.5 Key qualifier formats 22
 - 5.6 Primary identifier and value concatenation 23
 - 5.7 Key qualifier concatenation 23
 - 5.8 Path element order 24
 - 5.9 Data attributes 25
 - 5.9.1 Extension mechanism and reserved keywords 32
 - 5.9.2 Constructing the query string 32
 - 5.10 Constructing the GS1 Digital Link URI 33
 - 5.11 Canonical GS1 Digital Link URIs 34
- 6 Examples of GS1 Digital Link URIs 35**
 - 6.1 GTIN 35
 - 6.2 GTIN + CPV 36
 - 6.3 GTIN + Batch/Lot 36
 - 6.4 GTIN + Serial Number (also known as SGTIN) 36
 - 6.5 GTIN + Batch/Lot + Serial Number + Expiry Date 37
 - 6.6 GTIN + Net Weight 37
 - 6.7 SSCC 37
 - 6.8 SSCC with specified Content, Count and Batch/Lot 38
 - 6.9 Physical location represented by a GLN or GLN + GLN Extension 38
- 7 Resolving GS1 Digital Link URIs 39**
 - 7.1 Resolver Functionality 39
 - 7.2 Decompression 43
 - 7.3 Validation 43
 - 7.3.1 Validation detail 43
 - 7.4 Link metadata 46
 - 7.4.1 The target URL 46
 - 7.4.2 The link relation type (linkType) 46
 - 7.4.3 A title for the link 47
 - 7.4.4 The Media Type for the content 47

- 7.4.5 The human language of the linked resource 47
- 7.5 Provision of links in HTTP headers 47
- 7.6 Associating links with identified items 48
- 7.7 Redirection 48
 - 7.7.1 Pattern-based redirection..... 49
 - 7.7.2 Default linkType and default link (resolver side) 49
 - 7.7.3 Requesting a specific link type (client side) 51
 - 7.7.4 Requesting all available links..... 51
 - 7.7.5 The context keyword..... 55
 - 7.7.6 Recognising the user’s language and requested media type 55
 - 7.7.7 Default linkType set to all 55
 - 7.7.8 Handling the query string 56
- 7.8 Providing content directly..... 56
- 7.9 Supported link relation types 56
- 7.10 Link type maintenance..... 58
- 7.11 Resolver description file 59
 - 7.11.1 JSON schema 60
 - 7.11.2 Example 60
- 7.12 Conformance statement..... 61
- 7.13 GS1 Class Two resolver..... 62

8 Compression and decompression..... 63

- 8.1 Purpose of compression 63
- 8.2 Technical requirements extracted from the business requirements 63
- 8.3 Viability of types of compression techniques and opportunities for compression 63
- 8.4 Structure of the compressed string..... 64
 - 8.4.1 Conversion between binary and URI-safe base 64 alphabet..... 64
 - 8.4.2 Various defined tables for GS1 AIs (length, format) 65
 - 8.4.3 How each GS1 AI key : value pair is encoded in binary 68
 - 8.4.4 Length indicators and Encoding indicators..... 69
- 8.5 Optimised encoding of combinations of GS1 Application Identifiers..... 72
- 8.6 Examples of binary encoding of GS1 Application Identifiers 75
 - 8.6.1 GS1 Application Identifiers whose values are all-numeric and fixed-length 75
 - 8.6.2 GS1 Application Identifiers whose values are all-numeric and variable-length..... 75
 - 8.6.3 GS1 Application Identifiers whose values are alphanumeric and variable-length..... 76
 - 8.6.4 Optimisation using pre-defined sequences of GS1 Application Identifiers..... 77
- 8.7 Formal ABNF grammar for compressed GS1 Digital Link URIs 79
 - 8.7.1 Partially compressed GS1 Digital Link URIs 79
 - 8.7.2 Fully compressed GS1 Digital Link URIs 80
- 8.8 Compression procedure and flowcharts 81
- 8.9 Decompression procedure and flowcharts..... 96

9 Semantics..... 110

- 9.1 Exposing GS1 Digital Link semantics to the outside world.....110
- 9.2 Equivalence.....110
- 9.3 Information encoded within the URI111
- 9.4 Trailing slashes.....112
- 9.5 The identified resource and the applicability of attributes112
- 9.6 Subclass relationship116

9.7	Interpreting the URI query string	116
9.7.1	Determine if the RDF Subject is a unique instance identifier	116
9.7.2	Determine class relationships that can be inferred	118
9.7.3	Extract further relationships expecting data values	119
9.7.4	Extract subclass relationships	129
9.8	Worked examples.....	130
9.8.1	GTIN + CPV	130
9.8.2	GTIN + batch/lot + Serial number + expiry date	130
9.8.3	GTIN + Measured Weight.....	131
9.9	AIs defined as Web vocabulary terms	131
9.10	Link type semantics.....	133
10	Changes since version 1.0	133
10.1	Deferred to version 1.2.....	134
10.2	Changes since version 1.1.0	134
11	Glossary	134
12	References	138
A.1	Intellectual Property	141
A.1.1	Introduction and Disclaimer	141
A.1.2	Notices	141

1 Executive Summary

This section is informative

The GS1 system was originally designed in the early 1970s and has evolved to serve the needs of supply chains and is most relevant from manufacturer to checkout. In today's connected world, where consumers and business partners' expectations are informed by everyday experiences online, GS1 needs to make a step change into the future. As a result of this standard, it is possible to represent GS1 identification keys consistently within Web addresses as well as within barcodes containing Web addresses, such that a single identification approach can support both product identification for supply chain applications *and* a link to online material for consumer and business partner interactions. It's this dual functionality and enormous flexibility that is currently not possible when, for example, Brand Owners embed an unstructured Web page address in a QR Code®.

Consumers routinely seek product information on the Web, especially via mobile, and manufacturers may respond to this demand by adding a second machine readable barcode to the product. For retail products, this is usually a QR Code®¹ that encodes the complete address of a Web page from which product and marketing information is available.

The normative portions of this specification set forth compliance criteria for URI syntax, resolver functionality, compression/decompression, and semantics that are data carrier agnostic. The intent is to enable a diverse ecosystem of implementations in which a variety of data carriers, including optical barcodes, digital watermarks, and RF data carriers, may be used in conjunction with network services from many providers, leveraging the power of GS1 identifiers.

There are several elements that make up the GS1 Digital Link ecosystem, for example:

- World Wide Web addresses that follow a precisely-defined structure to include GS1 element strings.
- Scanners that are programmed to extract those GS1 element strings and pass them on in exactly the same way as they do now so that no changes to existing backend software are needed.
- Applications that can directly extract those Web addresses when included in a data carrier.
- Web servers programmed to redirect requests to relevant sources of information wherever they may be (these are known as resolvers as they 'resolve' GS1 identifiers).
- Web servers programmed such that when processing a URL, irrespective of its structure, the response conforms to the GS1 Digital Link standard.

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

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

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

2 Introduction

This section is informative

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

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

3 Definitions and namespaces

This section is normative

This standard introduces the following terms (terms used but defined elsewhere are provided in the glossary, starting in section [11](#)).

GS1 Digital Link is the title of this standard and of the overall ecosystem that it defines. It is a general term that refers to the expression of the GS1 System of identifiers on the World Wide Web.

A **GS1 Digital Link URI** is a Web URI that encodes one or more GS1 Application Identifiers and their value(s) according to the structure defined in this standard.

An **identified item** is anything identified by one or more GS1 element strings, including their encoding as a GS1 Digital Link URI, be it a product, location, organisation, asset or anything else.

A GS1 conformant **resolver** is a Web server that is able to process GS1 Digital Link URIs, redirecting or directly answering requests in accordance with this standard.

A **link type** (or `linkType`) defines the relationship between the identified item and the linked resource, such as a product information page, a warranty registration page, a related video etc. It does not specify the media type or any other characteristic of the linked resource as this information is provided by metadata describing the resource itself.

Throughout this document, the following prefixes and namespaces are used meaning that, for example, `gs1:pip` is equivalent to `https://gs1.org/voc/pip`.

Table 3-1 Prefixes and namespaces used in this document

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

3.1 Typographical conventions used in this document

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

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

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

3.2 Conformance to this standard

This standard comprises a number of discrete elements against which conformance can be asserted:

- Conformance with the GS1 Digital Link URI syntax, expressed using ABNF grammar in section [5](#)
- Conformance with the GS1 resolver functionality as detailed in section [7](#) and summarised in section [7.12](#)
- Conformance with the compression/decompression algorithm as detailed in the flowcharts that form the bulk of section [8](#)
- Conformance with the GS1 Digital Link semantics detailed in section [9](#).

There is no single conformance statement for the entirety of GS1 Digital Link which, as defined above, is a very general term. It is therefore inappropriate to make a formal claim of broad conformance to the standard without specifying the elements with which conformance is claimed.

- ✔ **NOTE:** This standard discusses complete URIs encoded in data carriers such as QR codes and NFC tags. The potential use of software to construct those URIs from components discovered through scans of, for example, UPC/EAN barcodes or GS1 DataMatrix symbols, is out of scope.

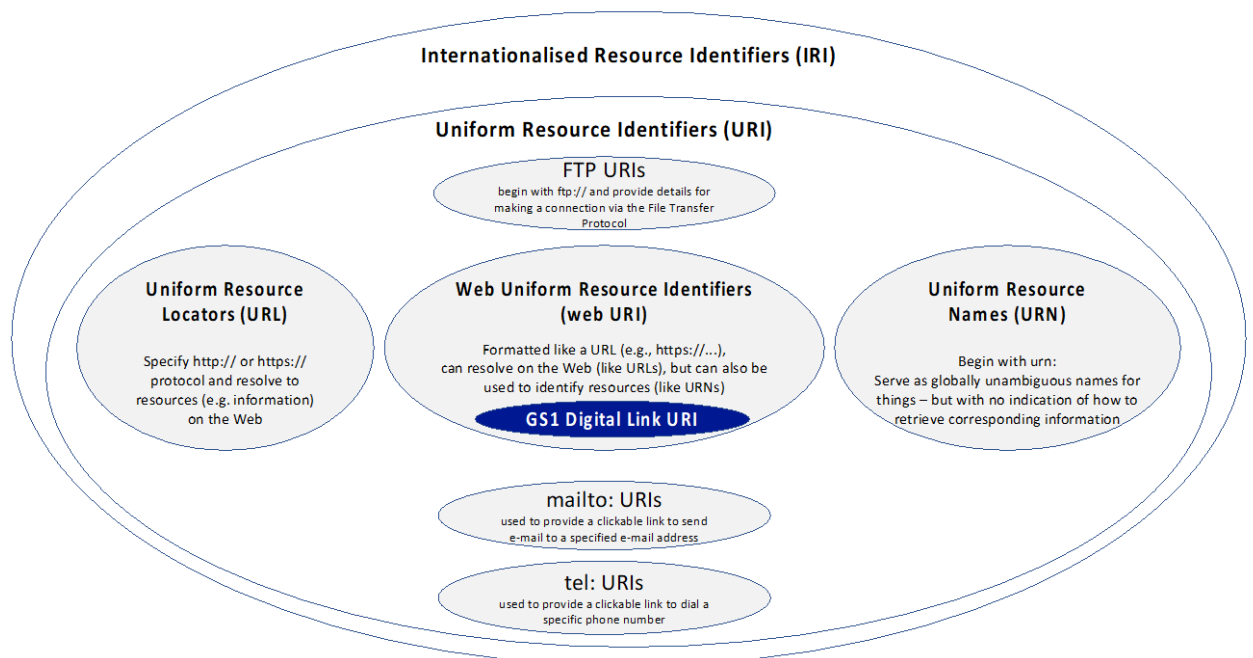
4 What is a URI?

This section is informative

Please note that this chapter, as well as chapters 5 and 6, have been superseded by GS1 Digital Link Standard: URI Syntax available from <https://ref.gs1.org/standards/digital-link/uri-syntax/>. All other sections are the current version of the standard.

This section provides some clarification about what a Uniform Resource Identifier (URI) is, how URIs relate to Uniform Resource Names (URNs) and Uniform Resource Locators (URLs), as well as providing an explanation of the main structural elements within a Web URI.

Figure 4-1 URNs and URLs are also URIs



[Figure 4-1](#) shows a Venn diagram in which we see that Uniform Resource Identifier is the broad term that includes Uniform Resource Names (URNs) and Uniform Resource Locators (URLs) as well as URIs with various protocols including http or https, ftp, mailto, tel etc. This means that every URL and every URN is also a URI, since URI is the broader umbrella term. Furthermore, Internationalized Resource Identifiers (IRIs) are an even broader category that support characters from the Universal Character Set/Unicode, whereas URIs only support the ASCII character set. IRIs are defined in [IRIs]. GS1 Digital Link URIs are a subset of Web URIs that conform to this GS1 technical standard.

[Figure 4-2](#) shows another Venn diagram. This time, it shows two capabilities:

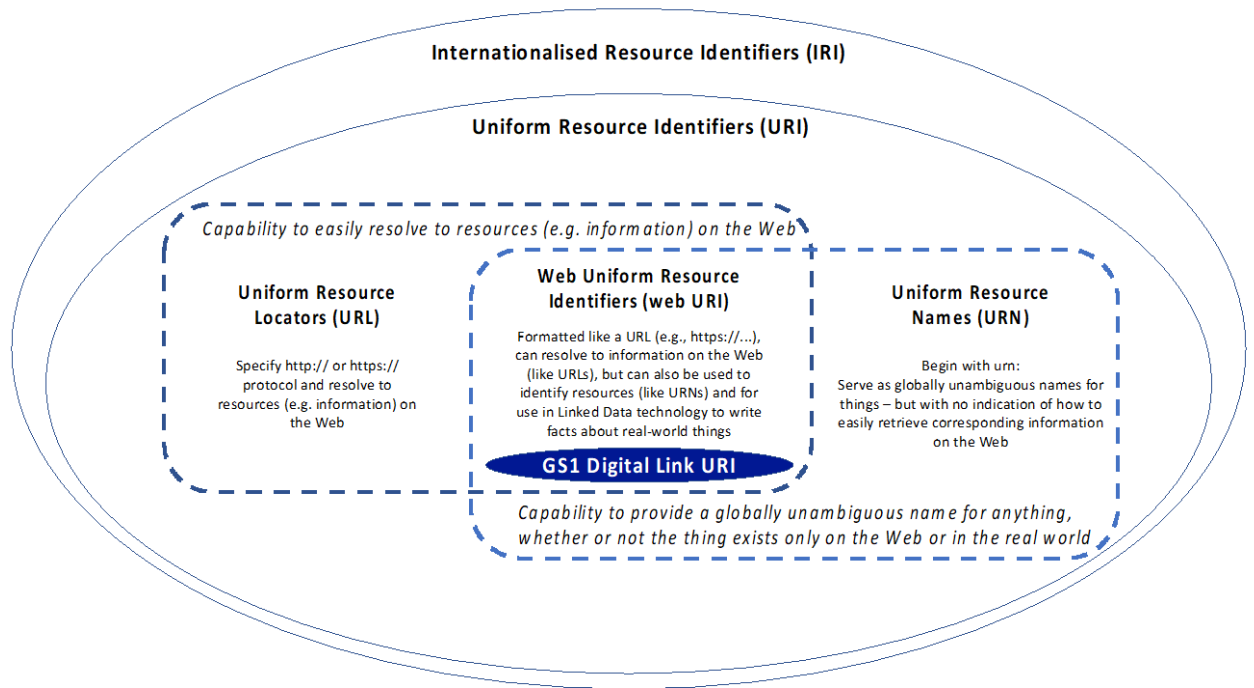
1. The capability to easily resolve to resources (e.g. information) on the Web.
2. The capability to provide a globally unambiguous name for anything, whether or not the thing exists only on the Web or in the real world.

The first capability is usually associated with URLs and Web addresses.

The second capability is usually associated with URNs.

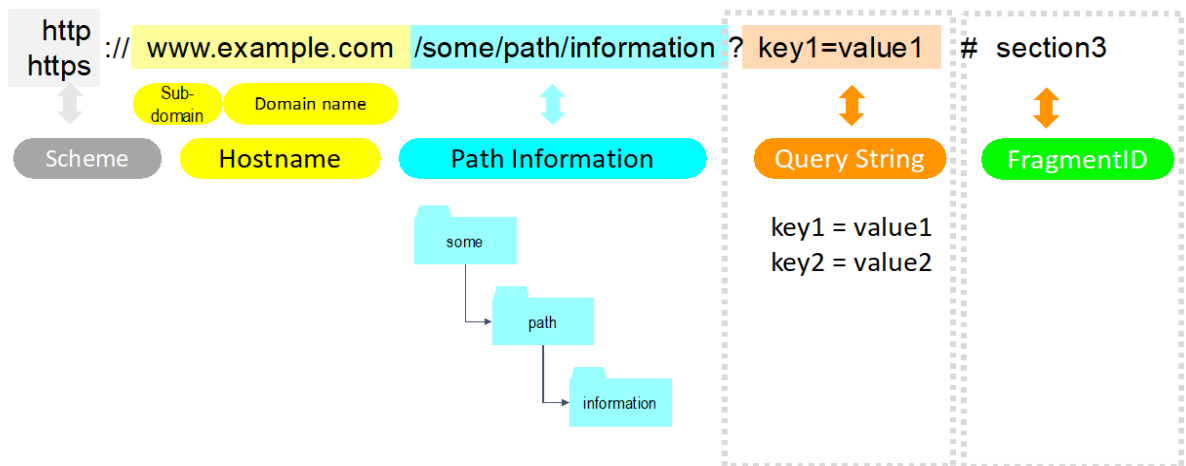
Web URIs exist at the intersection of these two capabilities; in terms of their syntax, they look like URLs because they specify http or https as their protocol - and they can be configured to behave like URLs in terms of supporting Web requests via the http/https Web protocol. However, they are also a perfectly valid way of assigning a globally unambiguous name for anything, whether in the real world or online. Note that 'globally unambiguous' does not mean globally unique; two different things should have distinct URIs in any situation where we want to be able to distinguish between them. However, there may be many URIs that all refer to the same thing, even within the same URI namespace or domain name. It is also possible to use Linked Data [Linked Data] to make an assertion between two URIs to formally express that they both refer to the same thing, even if the URIs are different strings.

Figure 4-2 A Web URI can act both as a globally unambiguous name for something, as well as providing an easy way to retrieve Web resources (e.g. information) relating to the identified thing



[Figure 4-3](#) provides a brief overview of the internal structural elements of a Web URI:

Figure 4-3 Internal structure of a Web URI



[Figure 4-3](#) shows the structural elements of a Web URI. The scheme indicates the protocol and (at the time of writing) is always `http://` or `https://` (use of HTTPS is more secure and is therefore recommended as best practice). The hostname is typically a registered Internet domain name or a subdomain of such a registered domain name. Following the domain name, the remainder of the Web URI is case sensitive. The URI path information consists of a number of strings separated by the forward slash character. Although this is just a string, it is often used by the Linked Data

community and in REST interfaces² to represent a collection of resources organised in a conceptually hierarchical way, with the broadest (most general, least specific) category appearing towards the left of the URI path information and with the narrowest (most specific) category appearing towards the right of the URI path information.

This design pattern provides a hint to humans that related Web URIs may exist and can be formed by successively truncating the Web URI path information from right to left, removing each successive segment preceded by its forward slash ("/") character. These related Web URIs may provide information about an object at a broader, more general, less specific granularity.

However, this is only a legible hint to humans. Computer software would typically treat the entire URI (at least up to the fragment identifier) as an opaque indivisible string and would not attempt such truncation. Instead, they will look for explicit links to related URIs, ideally expressed with semantic annotation, using Linked Data properties. These aspects – the machine-processable semantics or meaning of a GS1 Digital Link URI – are explored and defined in detail in section 9.

The query string enables multiple key=value pairs to be sent to a Web resource. The URI query string appears after the URI path information and consists of everything between the "?" at the end of the path information and the end of the URI or the "#" symbol indicating the start of the fragment identifier. Within the URI query string, key=value pairs may be concatenated using & or ; as a delimiter.

The URI fragment identifier is optional and appears after the query string (if present) and preceded by the "#" character. The URI fragment identifier is typically used to provide a link to an internal subsection of an information resource. The Linked Data community do make use of URIs with fragment identifiers, although the fragment identifier is not useful for passing key=value pairs. Importantly, fragment identifiers are *not* sent to the server but are handled entirely within the client.

Web URIs provide essentially two options for expressing the values of GS1 Application Identifiers - either within the URI path information or within the URI query string. The URI path information is the most appropriate place for expressing a GS1 identification key and an ordered set of optional qualifiers that are used in conjunction with the GS1 identification key to form a compound key that is used to retrieve information about something at a finer level of granularity (e.g. traceability data about an SGTIN, batch/lot-level master data). The query string is appropriate for data attributes of the identified resource such as expiry date, weight etc., as well as being a natural extension point for any additional arbitrary key=value pairs that cannot be expressed using GS1 Application Identifiers (see section 5.9.1); for example, the query string could include a key=value pair to indicate a specific stakeholder role or a specific action or activity or type of service to be accessed. It should be noted that no key=value pair should be repeated with the same key in the URI query string. If a key is repeated, the last defined value for that key takes precedence over any previously defined value.

4.1 The GS1 Digital Link URI

GS1 Digital Link provides a syntax for expressing GS1 identifier keys, key qualifiers and data attributes in a format that can be used on the Web in an intuitive manner (via a straightforward Web request) to enable consumers and others to directly access relevant information and services about products, assets, locations, etc. A GS1 Digital Link URI can be encoded natively in any data carrier that can support the encoding of a Web address (URL). This means that additional data carriers such as QR Codes®, digital watermarks, NFC tags and other technologies will also be able to include GS1 identification keys while continuing to provide links to relevant information. When the data carrier is created and such a URL is embedded within it, a scanning device can extract the entire URL, and no further processing by the scanning device, or software therein, is required to construct the URL that is used to access a server where relevant information is stored.

² See https://en.wikipedia.org/wiki/Representational_state_transfer

5 GS1 Digital Link URI Syntax

This section and all its subsections are normative

Please note that this chapter, as well as chapters 4 and 6, have been superseded by GS1 Digital Link Standard: URI Syntax available from <https://ref.gs1.org/standards/digital-link/uri-syntax/>. All other sections are the current version of the standard.

This section specifies the structure of GS1 Digital Link URIs using the Augmented Backus-Naur Form (ABNF) syntax as defined in [RFC 5234] and updated by [RFC 7405]. ABNF formally expresses how strings of characters (including URIs) are constructed by concatenating smaller components in a sequential order and is machine-processable.

Those smaller components may be defined in terms of further sub-components and/or in terms of sequences of character sets that are also defined by rules.

ABNF also supports repeating components and optional components. Optional components are enclosed within square brackets.

A sequential group of one or more components may be enclosed within round brackets.

Repeating components use the $m^*n(\text{component})$ notation to indicate that the component within the round brackets may appear at least m times and at most n times. Default values are $m=0$, $n=\text{infinity}$. If either or m or n are omitted, their default values are assumed.

Everything following a semicolon on a line is considered to be an explanatory comment.

The notation $n(\text{component})$ or $n\text{component}$ where n is one or more digit characters is equivalent to $n^*(\text{component})$, indicating that the component must appear exactly n times.

A number of comments are provided to explain the meaning of rules.

5.1 Character sets

Firstly, a number of character sets are defined for later re-use in subsequent ABNF rules.

```
DIGIT      = "0" / "1" / "2" / "3" / "4" /
            "5" / "6" / "7" / "8" / "9"
```

```
UPPERALPHA = %x41-5A ; A-Z ( ASCII characters 65-90 decimal, 41-5A hex)
```

```
LOWERALPHA = %x61-7A ; a-z ( ASCII characters 97-122 decimal, 61-7A hex)
```

```
ALPHA      = UPPERALPHA / LOWERALPHA ; A-Z or a-z
```

```
HEXDIG     = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
```

```
DoubleQuote = '"' ; the double-quote character "
```

The following characters must be represented using percent-encoding (see section 2.1 of RFC 3986 [PercentEncoding]) when used as literal characters within URIs, since many of these have special meanings within Web URIs:

```
Octothorpe      = "%x23" ; percent-encoding of the # character
```

```
ForwardSlash    = "%x2F" ; percent-encoding of the / character
```

Percent	= "%x25" ; percent-encoding of the % character
Ampersand	= "%x26" ; percent-encoding of the & character
Plus	= "%x2B" ; percent-encoding of the + character
Comma	= "%x2C" ; percent-encoding of the , character
Exclamation	= "%x21" ; percent-encoding of the ! character
LeftBracket	= "%x28" ; percent-encoding of the (character
RightBracket	= "%x29" ; percent-encoding of the) character
Asterisk	= "%x2A" ; percent-encoding of the * character
Apostrophe	= "%x27" ; percent-encoding of the ' character
Colon	= "%x3A" ; percent-encoding of the : character
Semicolon	= "%x3B" ; percent-encoding of the ; character
LeftAngleBracket	= "%x3C" ; percent-encoding of the < character
Equals	= "%x3D" ; percent-encoding of the = character
RightAngleBracket	= "%x3E" ; percent-encoding of the > character
QuestionMark	= "%x3F" ; percent-encoding of the ? character

The following group of symbol characters is permitted within the 82-character subset of ISO/IEC 646, indicated in Figure 7.11-1 of the GS1 General Specifications [GENSPECS].

```
XSYMBOL      = DoubleQuote / "-" / "." / "_" / Exclamation / Percent /
              Ampersand / Plus / Comma / ForwardSlash / Asterisk /
              LeftBracket / RightBracket / Apostrophe / Semicolon /
              Colon / LeftAngleBracket / RightAngleBracket / Equals /
              QuestionMark
```

The following group of symbol characters is permitted within the 39-character subset of ISO/IEC 646, indicated in Figure 7.11-2 of the GS1 General Specifications [GENSPECS].

```
YSYMBOL      = "-" / Octothorpe / ForwardSlash
```

The following character set corresponds to all permitted characters within the 82-character subset of ISO/IEC 646, indicated in Figure 7.11-1 of the GS1 General Specifications [GENSPECS].

```
XCHAR        = DIGIT / UPPERALPHA / LOWERALPHA / XSYMBOL
```

The following character set corresponds to all permitted characters within the 39-character subset of ISO/IEC 646, indicated in Figure 7.11-2 of the GS1 General Specifications [GENSPECS]. It is currently only used within the value of the Components and Parts Identifier (CPID).

```
YCHAR        = DIGIT / UPPERALPHA / YSYMBOL
```

5.2 Primary identification keys

The following rules indicate which GS1 Application Identifiers (AI) are considered as primary identification keys for GS1 Digital Link URI. Note that for each of these (and the rules in section

[5.3](#)), the numeric AI value may be used or alternatively, a corresponding lower-case short name may be used if it is more friendly to software developers. The numeric AI value may be more suitable for use when encoding a GS1 Digital Link URI within a 2D barcode, since this can be encoded more efficiently, resulting in a lower total module count and improved readability.

The `%s` prefix notation was introduced in [RFC 7405] and simply indicates that the following string value is case-sensitive. For example, in the rule below, `gtin-code` may be either `"01"` or `"gtin"` but not `"GTIN"` nor `"Gtin"`.

```

gtin-code           = "01" / %s"gtin"           ; GTIN
itip-code           = "8006" / %s"itip"          ; ITIP
gmn-code            = "8013" / %s"gmn"           ; Global Model Number
cpid-code           = "8010" / %s"cpid"          ; CPID
shipTo-code         = "410" / %s"shipTo"         ; ship-to
billTo-code         = "411" / %s"billTo"         ; bill-to
purchasedFrom-code = "412" / %s"purchasedFrom"   ; purchased from GLN
shipFor-code        = "413" / %s"shipFor"        ; ship-for
gln-code            = "414" / %s"gln"           ; Physical Location GLN
payTo-code          = "415" / %s"payTo"          ; GLN of invoicing party
glnProd-code        = "416" / %s"glnProd"        ; GLN of production/service loc.
partyGln-code       = "417" / %s"party"         ; Party GLN
gsrnp-code          = "8017" / %s"gsrnp"         ; GSRN of the Provider
gsrn-code           = "8018" / %s"gsrn"         ; GSRN of the Recipient
gcn-code            = "255" / %s"gcn"           ; GCN
sscc-code           = "00" / %s"sscc"          ; SSCC
gdti-code           = "253" / %s"gdti"          ; GDTI
ginc-code           = "401" / %s"ginc"          ; GINC
gsin-code           = "402" / %s"gsin"          ; GSIN
grai-code           = "8003" / %s"grai"         ; GRAI
giai-code           = "8004" / %s"giai"         ; GIAI
  
```

5.3 Key qualifiers

The following rules which GS1 Application Identifiers (AI) are considered as key qualifiers for a GS1 Digital Link URI.

```

cpv-code           = "22" / %s"cpv"           ; Consumer Product Variant
lot-code           = "10" / %s"lot"          ; Batch/Lot identifier
ser-code           = "21" / %s"ser"          ; GTIN Serial Number
cpsn-code          = "8011" / %s"cpsn"        ; CPID Serial Number
glnx-code          = "254" / %s"glnx"        ; GLN extension
refno-code         = "8020" / %s"refno"       ; Payment Reference Number
srin-code          = "8019" / %s"srin"        ; Service Relation Instance Number
tpx-code           = "235"
                   ; third-party controlled serialised extension to GTIN
  
```

```
uic-ext-code          = "7040"
    ; GS1 UIC with Extension 1 and Importer Index
```

5.4 Primary key formats

The following rules express the format of the values of the primary GS1 identification keys.



Note: the GS1 General Specifications [GENSPECS] define further restrictions on some of these values, particularly for those which include a GS1 Check Digit, Indicator Digit or Extension Digit. Please refer to the GS1 General Specifications [GENSPECS] for further details.

```
gtin-value           = 8DIGIT / 12DIGIT / 13DIGIT / 14DIGIT
itip-value           = 14DIGIT 2DIGIT 2DIGIT
    ; 14 digits then 2 digits then 2 digits
gmn-value            = 1*30XCHAR ; 1-30 characters from 82-chr subset
cpid-value           = 1*30YCHAR ; 1-30 characters from 39-chr subset
shipTo-value         = 13DIGIT ; exactly 13 digits
billTo-value         = 13DIGIT ; exactly 13 digits
purchasedFrom-value = 13DIGIT ; exactly 13 digits
shipFor-value        = 13DIGIT ; exactly 13 digits
gln-value            = 13DIGIT ; exactly 13 digits
payTo-value          = 13DIGIT ; exactly 13 digits
glnProd-value        = 13DIGIT ; exactly 13 digits
partyGln-value       = 13DIGIT ; exactly 13 digits
gsrnp-value          = 18DIGIT ; exactly 18 digits
gsrn-value           = 18DIGIT ; exactly 18 digits
gcn-value            = 13DIGIT 1*12DIGIT ; 13 digits then 1-12 digits
sscc-value           = 18DIGIT ; exactly 18 digits
gdtdi-value          = 13DIGIT 1*17XCHAR
    ; 13 digits then 1-17 characters
    ; from the 82-character subset
ginc-value           = 1*30XCHAR
    ; 1-30 characters from the 82-character subset
gsin-value           = 17DIGIT ; exactly 17 digits
grai-value           = 14DIGIT 1*16XCHAR
    ; 14 digits then 1-16 characters
    ; from the 82-character subset of ISO/IEC 646
giai-value           = 1*30XCHAR ; 1-30 characters from 82-chr subset
```

5.5 Key qualifier formats

The following rules express the format of the values of the key qualifiers of primary GS1 identification keys:

```
cpv-value            = 1*20XCHAR ; 1-20 characters from 82-chr subset
```

lot-value	= 1*20XCHAR ; 1-20 characters from 82-chr subset
ser-value	= 1*20XCHAR ; 1-20 characters from 82-chr subset
cpsn-value	= 1*12DIGIT ; 1-12 digits
glnx-value	= 1*20XCHAR ; 1-20 characters from 82-chr subset
refno-value	= 1*25XCHAR ; 1-25 characters from 82-chr subset
srin-value	= 1*10DIGIT ; 1-10 digits
tpx-value	= 1*28XCHAR ; 1-28 characters from 82-chr subset
uic-ext-value	= 1DIGIT 3XCHAR ; 1 digit then 3 characters from 82-chr subset

5.6 Primary identifier and value concatenation

The following rules express how each primary identifier code and its value should be concatenated (for use within the URI path information) :

gtin-comp	= "/" gtin-code "/" gtin-value
itip-comp	= "/" itip-code "/" itip-value
gmn-comp	= "/" gmn-code "/" gmn-value
cpid-comp	= "/" cpid-code "/" cpid-value
shipTo-comp	= "/" shipTo-code "/" shipTo-value
billTo-comp	= "/" billTo-code "/" billTo-value
purchasedFrom-comp	= "/" purchasedFrom-code "/" purchasedFrom-value
shipFor-comp	= "/" shipFor-code "/" shipFor-value
gln-comp	= "/" gln-code "/" gln-value
payTo-comp	= "/" payTo-code "/" payTo-value
glnProd-comp	= "/" glnProd-code "/" glnProd-value
partyGln-comp	= "/" partyGln-code "/" partyGln-value
gsrnp-comp	= "/" gsrnp-code "/" gsrnp-value
gsrn-comp	= "/" gsrn-code "/" gsrn-value
gcn-comp	= "/" gcn-code "/" gcn-value
sscc-comp	= "/" sccc-code "/" sccc-value
gdti-comp	= "/" gdti-code "/" gdti-value
ginc-comp	= "/" ginc-code "/" ginc-value
gsin-comp	= "/" gsin-code "/" gsin-value
grai-comp	= "/" grai-code "/" grai-value
giai-comp	= "/" giai-code "/" giai-value

5.7 Key qualifier concatenation

The following rules express how each key qualifier and its value should be concatenated (for use within the URI path information) :

cpv-comp	= "/" cpv-code "/" cpv-value
lot-comp	= "/" lot-code "/" lot-value
ser-comp	= "/" ser-code "/" ser-value

```

cpsn-comp           = "/" cpsn-code "/" cpsn-value
glnx-comp           = "/" glnx-code "/" glnx-value
refno-comp          = "/" refno-code "/" refno-value
srin-comp           = "/" srin-code "/" srin-value
tpx-comp            = "/" tpx-code "/" tpx-value
uic-ext-comp        = "/" uic-ext-code "/" uic-ext-value
  
```

5.8 Path element order

The following rules express how the URI path information should be structured for each primary GS1 identification key. Note that some primary identifiers such as SSCC do not have any associated key qualifier. Other primary identifiers such as GTIN may have multiple key qualifiers. The square bracket notation indicates that the enclosed key qualifier component may be omitted but the sequence in which they appear is important and must be preserved. For example, the rule for gtin-path would permit any of these:

```

/gtin/01234567890128/cpv/2A/lot/ABC123/ser/12345XYZ
/gtin/01234567890128/lot/ABC123/
/gtin/01234567890128/lot/ABC123/ser/12345XYZ
/gtin/01234567890128/ser/12345XYZ
  
```

but does not permit strings such as:

```

/gtin/01234567890128/ser/12345XYZ/lot/ABC123
  
```

in which the sequential ordering of the key qualifier components is not preserved.

```

gtin-path           = gtin-comp [cpv-comp] [lot-comp] [ser-comp]
itip-path           = itip-comp [cpv-comp] [lot-comp] [ser-comp]
gmn-path            = gmn-comp
cpid-path           = cpid-comp [cpsn-comp]
shipTo-path         = shipTo-comp
billTo-path         = billTo-comp
purchasedFrom-path = purchasedFrom-comp
shipFor-path        = shipFor-comp
gln-path            = gln-comp [glnx-comp]
payTo-path          = payTo-comp
glnProd-path        = glnProd-comp
partyGln-path       = partyGln-comp
gsrnp-path          = gsrnp-comp [srin-comp]
gsrn-path           = gsrn-comp [srin-comp]
gcn-path            = gcn-comp
sscc-path           = sccc-comp
gdti-path           = gdti-comp
ginc-path           = ginc-comp
gsin-path           = gsin-comp
grai-path           = grai-comp
giai-path           = giai-comp
upui-path           = gtin-comp tpx-comp
eoid-path           = partyGln-comp uic-ext-comp
fid-path            = gln-comp uic-ext-comp
mid-path            = giai-comp uic-ext-comp
  
```


The following rule simply states that any of the above is considered as a `gs1path` (which will be referenced in a later rule).

```
gs1path = gtin-path / itip-path / gmn-path / cpid-path / shipTo-path /
billTo-path / purchasedFrom-path / shipFor-path / gln-path /
payTo-path / glnProd-path / partyGln-path / gsrnp-path /
gsrn-path / gcn-path / ssc-path / gdti-path / ginc-path /
gsin-path / grai-path / giai-path / upui-path / eoid-path /
fid-path / mid-path
```

5.9 Data attributes

The following rules are concerned with GS1 Application Identifiers that are considered to be data attributes rather than primary identifier keys or key qualifiers. Data attributes and their values SHALL be expressed via the URI query string as `key=value` pairs. For most of these, the key is always the numeric AI value, rather than a more human-friendly short name.

```
netWeightVMTICode = "3100" / "3101" / "3102" / "3103" / "3104" / "3105" /
"3200" / "3201" / "3202" / "3203" / "3204" / "3205" /
"3560" / "3561" / "3562" / "3563" / "3564" / "3565" /
"3570" / "3571" / "3572" / "3573" / "3574" / "3575"
netWeightVMTIValue = 6DIGIT
netWeightVMTIPParameter = netWeightVMTICode "=" netWeightVMTIValue

lengthVMTICode = "3110" / "3111" / "3112" / "3113" / "3114" / "3115" /
"3210" / "3211" / "3212" / "3213" / "3214" / "3215" /
"3220" / "3221" / "3222" / "3223" / "3224" / "3225" /
"3230" / "3231" / "3232" / "3233" / "3234" / "3235"
lengthVMTIValue = 6DIGIT
lengthVMTIPParameter = lengthVMTICode "=" lengthVMTIValue

widthVMTICode = "3120" / "3121" / "3122" / "3123" / "3124" / "3125" /
"3240" / "3241" / "3242" / "3243" / "3244" / "3245" /
"3250" / "3251" / "3252" / "3253" / "3254" / "3255" /
"3260" / "3261" / "3262" / "3263" / "3264" / "3265"
widthVMTIValue = 6DIGIT
widthVMTIPParameter = widthVMTICode "=" widthVMTIValue

depthVMTICode = "3130" / "3131" / "3132" / "3133" / "3134" / "3135" /
"3270" / "3271" / "3272" / "3273" / "3274" / "3275" /
"3280" / "3281" / "3282" / "3283" / "3284" / "3285" /
"3290" / "3291" / "3292" / "3293" / "3294" / "3295"
depthVMTIValue = 6DIGIT
depthVMTIPParameter = depthVMTICode "=" depthVMTIValue

areaVMTICode = "3140" / "3141" / "3142" / "3143" / "3144" / "3145" /
"3500" / "3501" / "3502" / "3503" / "3504" / "3505" /
"3510" / "3511" / "3512" / "3513" / "3514" / "3515" /
"3520" / "3521" / "3522" / "3523" / "3524" / "3525"
areaVMTIValue = 6DIGIT
areaVMTIPParameter = areaVMTICode "=" areaVMTIValue

netVolumeVMTICode = "3150" / "3151" / "3152" / "3153" / "3154" / "3155" /
"3160" / "3161" / "3162" / "3163" / "3164" / "3165" /
"3600" / "3601" / "3602" / "3603" / "3604" / "3605" /
"3610" / "3611" / "3612" / "3613" / "3614" / "3615" /
"3640" / "3641" / "3642" / "3643" / "3644" / "3645" /
"3650" / "3651" / "3652" / "3653" / "3654" / "3655" /
"3660" / "3661" / "3662" / "3663" / "3664" / "3665"
netVolumeVMTIValue = 6DIGIT
netVolumeVMTIPParameter = netVolumeVMTICode "=" netVolumeVMTIValue
```

```

massPerUnitAreaVMTICode    = "3370" / "3371" / "3372" / "3373" / "3374" /
                             "3375"
massPerUnitAreaVMTIValue   = 6DIGIT
massPerUnitAreaVMTIParameter = massPerUnitAreaVMTICode "="
                             massPerUnitAreaVMTIValue

grossWeightCode           = "3300" / "3301" / "3302" / "3303" / "3304" / "3305" /
                             "3400" / "3401" / "3402" / "3403" / "3404" / "3405"
grossWeightValue          = 6DIGIT
grossWeightParameter      = grossWeightCode "=" grossWeightValue

logisticLengthCode        = "3310" / "3311" / "3312" / "3313" / "3314" / "3315" /
                             "3410" / "3411" / "3412" / "3413" / "3414" / "3415" /
                             "3420" / "3421" / "3422" / "3423" / "3424" / "3425" /
                             "3430" / "3431" / "3432" / "3433" / "3434" / "3435"
logisticLengthValue       = 6DIGIT
logisticLengthParameter   = logisticLengthCode "=" logisticLengthValue

logisticWidthCode         = "3320" / "3321" / "3322" / "3323" / "3324" / "3325" /
                             "3440" / "3441" / "3442" / "3443" / "3444" / "3445" /
                             "3450" / "3451" / "3452" / "3453" / "3454" / "3455" /
                             "3460" / "3461" / "3462" / "3463" / "3464" / "3465"
logisticWidthValue        = 6DIGIT
logisticWidthParameter    = logisticWidthCode "=" logisticWidthValue

logisticDepthCode         = "3330" / "3331" / "3332" / "3333" / "3334" / "3335" /
                             "3470" / "3471" / "3472" / "3473" / "3474" / "3475" /
                             "3480" / "3481" / "3482" / "3483" / "3484" / "3485" /
                             "3490" / "3491" / "3492" / "3493" / "3494" / "3495"
logisticDepthValue        = 6DIGIT
logisticDepthParameter    = logisticDepthCode "=" logisticDepthValue

logisticAreaCode          = "3340" / "3341" / "3342" / "3343" / "3344" / "3345" /
                             "3530" / "3531" / "3532" / "3533" / "3534" / "3535" /
                             "3540" / "3541" / "3542" / "3543" / "3544" / "3545" /
                             "3550" / "3551" / "3552" / "3553" / "3554" / "3555"
logisticAreaValue         = 6DIGIT
logisticAreaParameter     = logisticAreaCode "=" logisticAreaValue

logisticVolumeCode        = "3350" / "3351" / "3352" / "3353" / "3354" / "3355" /
                             "3360" / "3361" / "3362" / "3363" / "3364" / "3365" /
                             "3620" / "3621" / "3622" / "3623" / "3624" / "3625" /
                             "3630" / "3631" / "3632" / "3633" / "3634" / "3635" /
                             "3670" / "3671" / "3672" / "3673" / "3674" / "3675" /
                             "3680" / "3681" / "3682" / "3683" / "3684" / "3685" /
                             "3690" / "3691" / "3692" / "3693" / "3694" / "3695"
logisticVolumeValue       = 6DIGIT
logisticVolumeParameter   = logisticVolumeCode "=" logisticVolumeValue

processorCode              = "7030" / "7031" / "7032" / "7033" / "7034" / "7035" /
                             "7036" / "7037" / "7038" / "7039"
processorValue             = 3DIGIT 1*27XCHAR
processorParameter         = processorCode "=" processorValue

contentCode                = "02"
contentValue               = 14DIGIT
contentParameter          = contentCode "=" contentValue
  
```

```
prodDateCode           = "11"  
prodDateValue          = 6DIGIT  
prodDateParameter     = prodDateCode "=" prodDateValue  
  
dueDateCode           = "12"  
dueDateValue          = 6DIGIT  
dueDateParameter     = dueDateCode "=" dueDateValue  
  
packDateCode          = "13"  
packDateValue         = 6DIGIT  
packDateParameter    = packDateCode "=" packDateValue  
  
bestBeforeDateCode   = "15"  
bestBeforeDateValue  = 6DIGIT  
bestBeforeDateParameter = bestBeforeDateCode "=" bestBeforeDateValue  
  
sellByDateCode       = "16"  
sellByDateValue      = 6DIGIT  
sellByDateParameter  = sellByDateCode "=" sellByDateValue  
  
firstFreezeDateCode  = "7006"  
firstFreezeDateValue = 6DIGIT  
firstFreezeDateParameter = firstFreezeDateCode "=" firstFreezeDateValue  
  
harvestDateCode      = "7007"  
harvestDateValue     = 6*12DIGIT  
harvestDateParameter = harvestDateCode "=" harvestDateValue  
  
pricePerUnitCode     = "8005"  
pricePerUnitValue    = 6DIGIT  
pricePerUnitParameter = pricePerUnitCode "=" pricePerUnitValue  
  
variantCode          = "20"  
variantValue         = 2DIGIT  
variantParameter     = variantCode "=" variantValue  
  
varCountCode         = "30"  
varCountValue        = 1*8DIGIT  
varCountParameter    = varCountCode "=" varCountValue  
  
countCode            = "37"  
countValue           = 1*8DIGIT  
countParameter       = countCode "=" countValue  
  
mutualCode           = "90"  
mutualValue          = 1*30DIGIT  
mutualParameter      = mutualCode "=" mutualValue  
  
additionalIdCode     = "240"  
additionalIdValue    = 1*30DIGIT  
additionalIdParameter = additionalIdCode "=" additionalIdValue  
  
custPartNoCode       = "241"  
custPartNoValue      = 1*30DIGIT  
custPartNoParameter  = custPartNoCode "=" custPartNoValue  
  
mtoVariantCode       = "242"  
mtoVariantValue      = 6DIGIT  
mtoVariantParameter  = mtoVariantCode "=" mtoVariantValue
```

```
pcnCode           = "243"
pcnValue          = 1*20DIGIT
pcnParameter      = pcnCode "=" pcnValue

secondarySerialCode = "250"
secondarySerialValue = 1*30DIGIT
secondarySerialParameter = secondarySerialCode "=" secondarySerialValue

refToSourceCode   = "251"
refToSourceValue  = 1*30DIGIT
refToSourceParameter = refToSourceCode "=" refToSourceValue

amountCode        = "3900" / "3901" / "3902" / "3903" / "3904" / "3905"
amountValue       = 1*15DIGIT
amountParameter   = amountCode "=" amountValue

amountISOCCode    = "3910" / "3911" / "3912" / "3913" / "3914" / "3915"
amountISOValue    = 3DIGIT 1*15DIGIT
amountISOPParameter = amountISOCCode "=" amountISOValue

priceCode         = "3920" / "3921" / "3922" / "3923" / "3924" / "3925"
priceValue        = 1*15DIGIT
priceParameter    = priceCode "=" priceValue

priceISOCCode     = "3930" / "3931" / "3932" / "3933" / "3934" / "3935"
priceISOValue     = 3DIGIT 1*15DIGIT
priceISOPParameter = priceISOCCode "=" priceISOValue

percentOffCode    = "3940" / "3941" / "3942" / "3943" / "3944" / "3945"
percentOffValue   = 4DIGIT
percentOffParameter = percentOffCode "=" percentOffValue

orderNumberCode   = "400"
orderNumberValue  = 1*30DIGIT
orderNumberParameter = orderNumberCode "=" orderNumberValue

routeCode         = "403"
routeValue        = 1*30DIGIT
routeParameter    = routeCode "=" routeValue

shipToLocCode     = "410"
shipToLocValue    = 13DIGIT
shipToLocParameter = shipToLocCode "=" shipToLocValue

billToCode        = "411"
billToValue       = 13DIGIT
billToParameter   = billToCode "=" billToValue

purchaseFromCode  = "412"
purchaseFromValue = 13DIGIT
purchaseFromParameter = purchaseFromCode "=" purchaseFromValue

shipForLocCode    = "413"
shipForLocValue   = 13DIGIT
shipForLocParameter = shipForLocCode "=" shipForLocValue

locNoCode         = "414"
locNoValue        = 13DIGIT
locNoParameter    = locNoCode "=" locNoValue
```

```

payToCode           = "415"
payToValue          = 13DIGIT
payToParameter      = payToCode "=" payToValue

prodServLocCode     = "416"
prodServLocValue    = 13DIGIT
prodServLocParameter = prodServLocCode "=" prodServLocValue

shipToPostCode      = "420"
shipToPostValue     = 1*20XCHAR
shipToPostParameter = shipToPostCode "=" shipToPostValue

shipToPostISOCODE   = "421"
shipToPostISOValue  = 3DIGIT 1*9XCHAR
shipToPostISOParameter = shipToPostISOCODE "=" shipToPostISOValue

originCode           = "422"
originValue          = 3DIGIT
originParameter      = originCode "=" originValue

countryProcessCode  = "424"
countryProcessValue  = 3DIGIT
countryProcessParameter = countryProcessCode "=" countryProcessValue

countryFullProcessCode = "426"
countryFullProcessValue = 3DIGIT
countryFullProcessParameter =
    countryFullProcessCode "=" countryFullProcessValue

countryInitialProcessCode = "423"
countryInitialProcessValue = 3DIGIT 1*12DIGIT
countryInitialProcessParameter =
    countryInitialProcessCode "=" countryInitialProcessValue

countryDisassemblyCode = "425"
countryDisassemblyValue = 3DIGIT 1*12DIGIT
countryDisassemblyParameter =
    countryDisassemblyCode "=" countryDisassemblyValue

originSubdivisionCode = "427"
originSubdivisionValue = 1*3XCHAR
originSubdivisionParameter =
    originSubdivisionCode "=" originSubdivisionValue

nhrnPZNCode         = "710"
nhrnPZNValue        = 1*20XCHAR
nhrnPZNParameter     = nhrnPZNCode "=" nhrnPZNValue

nhrnCIPCode         = "711"
nhrnCIPValue        = 1*20XCHAR
nhrnCIPParameter    = nhrnCIPCode "=" nhrnCIPValue

nhrnCNCode          = "712"
nhrnCNValue         = 1*20XCHAR
nhrnCNParameter     = nhrnCNCode "=" nhrnCNValue

nhrnDRNCode         = "713"
nhrnDRNValue        = 1*20XCHAR
nhrnDRNParameter    = nhrnDRNCode "=" nhrnDRNValue
  
```

nhrnAIMCode	= "714"
nhrnAIMValue	= 1*20XCHAR
nhrnAIMParameter	= nhrnAIMCode "=" nhrnAIMValue
nsnCode	= "7001"
nsnValue	= 13DIGIT
nsnParameter	= nsnCode "=" nsnValue
meatCutCode	= "7002"
meatCutValue	= 1*30XCHAR
meatCutParameter	= meatCutCode "=" meatCutValue
activePotencyCode	= "7004"
activePotencyValue	= 1*4DIGIT
activePotencyParameter	= activePotencyCode "=" activePotencyValue
catchAreaCode	= "7005"
catchAreaValue	= 1*12XCHAR
catchAreaParameter	= catchAreaCode "=" catchAreaValue
aquaticSpeciesCode	= "7008"
aquaticSpeciesValue	= 1*3XCHAR
aquaticSpeciesParameter	= aquaticSpeciesCode "=" aquaticSpeciesValue
fishingGearTypeCode	= "7009"
fishingGearTypeValue	= 1*10XCHAR
fishingGearTypeParameter	= fishingGearTypeCode "=" fishingGearTypeValue
prodMethodCode	= "7010"
prodMethodValue	= 1*2XCHAR
prodMethodParameter	= prodMethodCode "=" prodMethodValue
refurbLotCode	= "7020"
refurbLotValue	= 1*20XCHAR
refurbLotParameter	= refurbLotCode "=" refurbLotValue
funcStatCode	= "7021"
funcStatValue	= 1*20XCHAR
funcStatParameter	= funcStatCode "=" funcStatValue
revStatCode	= "7022"
revStatValue	= 1*20XCHAR
revStatParameter	= revStatCode "=" revStatValue
giaiAssemblyCode	= "7023"
giaiAssemblyValue	= 1*30XCHAR
giaiAssemblyParameter	= giaiAssemblyCode "=" giaiAssemblyValue
certificationRefCode	= "7230" / "7231" / "7232" / "7233" / "7234" / "7235" / "7236" / "7237" / "7238" / "7239"
certificationRefValue	= 2*30XCHAR
certificationRefParameter	= certificationRefCode "=" certificationRefValue
dimensionsCode	= "8001"
dimensionsValue	= 14DIGIT
dimensionsParameter	= dimensionsCode "=" dimensionsValue
cmtNoCode	= "8002"

```

cmtNoValue           = 1*20XCHAR
cmtNoParameter       = cmtNoCode "=" cmtNoValue

ibanCode             = "8007"
ibanValue            = 1*34XCHAR
ibanParameter        = ibanCode "=" ibanValue

prodTimeCode         = "8008"
prodTimeValue        = 8DIGIT 1*4DIGIT
prodTimeParameter    = prodTimeCode "=" prodTimeValue

opticalSensorCode    = "8009"
opticalSensorValue   = 1*50XCHAR
opticalSensorParameter = opticalSensorCode "=" opticalSensorValue

versionCode          = "8012"
versionValue         = 4DIGIT 1*20XCHAR
versionParameter     = versionCode "=" versionValue

refNoCode            = "8020"
refNoValue           = 1*25XCHAR
refNoParameter       = refNoCode "=" refNoValue

itipContentCode      = "8026"
itipContentValue     = 14DIGIT 2DIGIT 2DIGIT
itipContentParameter = itipContentCode "=" itipContentValue

couponIDNACode       = "8110"
couponIDNAValue      = 1*70XCHAR
couponIDNAPParameter = couponIDNACode "=" couponIDNAValue

pointsCode           = "8111"
pointsValue          = 4DIGIT
pointsParameter      = pointsCode "=" pointsValue

paperlessCouponIDNACode = "8112"
paperlessCouponIDNAValue = 1*70XCHAR
paperlessCouponIDNAPParameter =
    paperlessCouponIDNACode "=" paperlessCouponIDNAValue

internalCode         = "91" / "92" / "93" / "94" / "95" /
                    "96" / "97" / "98" / "99"
internalValue        = 1*90XCHAR
internalParameter    = internalCode "=" internalValue
  
```

Batch/Lot may also be used as a data attribute in conjunction with an SSCC [AI (00)] and a CONTENT [AI (02)] in order to indicate that the SSCC contains GTINs of a specific batch/lot. For this reason, `LotParameter` is defined for use within the URI query string.

```
LotParameter      = lot-code "=" lot-value
```

Expiry Date [AI (17)] and Expiry Date/Time [AI (7003)] are data attributes. However, because of their importance in managing stock rotation and checking for expired products, the following rules also define a lower-case short name, `exp` and `expdt` that may be used in place of numeric AIs "17" and "7003" respectively.

```
expiryDateCode    = "17" / %s"exp"
```

```

expiryDateValue      = 6DIGIT
expiryDateParameter  = expiryDateCode "=" expiryDateValue

expiryTimeCode       = "7003" / %s"expdt"
expiryTimeValue      = 10DIGIT
expiryTimeParameter  = expiryTimeCode "=" expiryTimeValue
  
```

5.9.1 Extension mechanism and reserved keywords

The URI query string is a natural extension point within the syntax that can accommodate additional key=value pairs to express data attribute parameters that cannot be expressed using GS1 Application Identifiers. Examples of such usage may be to express a specific role, action, activity or type of service to be accessed. The following `extensionParameter` is based on the ABNF rule for query appearing in [RFC 3986] and serves as the main extension point for the GS1 Digital Link URI syntax. It permits multiple arbitrary key=value pairs to be included within the query string of a GS1 Digital Link URI. Any key=value pairs used for extension data SHALL NOT be all-numeric to avoid conflict with existing and future keys used for GS1 Application Identifiers either in terms of semantics or syntax; and SHALL NOT use the values `lot`, `exp`, `expdt`; nor should they be used to express a value (such as a value for net weight) if that value can be expressed using GS1 Application Identifiers as data attributes. As detailed in sections [7.7.3](#) and [7.7.5](#), the keywords `linkType` and `context` are also reserved and SHALL NOT be used except as defined in those sections.

```

extensionParameter    = *( pchar / "/" / "?" )
                        ; any other query string parameter permitted by RFC 3986
                        ; including additional arbitrary key=value pairs except as
                        ; restricted in the above paragraph
  
```

5.9.2 Constructing the query string

The following rule states that any of the above parameters for data attributes may appear as a query string parameter (`queryStringParam`), referenced later.

```

queryStringParam      = netWeightVMTIPParameter / lengthVMTIPParameter /
                        widthVMTIPParameter / depthVMTIPParameter / areaVMTIPParameter /
                        netVolumeVMTIPParameter / massPerUnitAreaVMTIPParameter /
                        grossWeightParameter / logisticLengthParameter /
                        logisticWidthParameter / logisticDepthParameter /
                        logisticAreaParameter / logisticVolumeParameter /
                        processorParameter / LotParameter / expiryDateParameter /
                        expiryTimeParameter / contentParameter / prodDateParameter /
                        dueDateParameter / packDateParameter / bestBeforeDateParameter /
                        sellByDateParameter / firstFreezeDateParameter /
                        harvestDateParameter / pricePerUnitParameter / variantParameter /
                        varCountParameter / countParameter / internalParameter /
                        additionalIdParameter / custPartNoParameter /
                        mtoVariantParameter / pcnParameter / secondarySerialParameter /
                        refToSourceParameter / amountParameter / amountISOParameter /
                        priceParameter / priceISOParameter / percentOffParameter /
                        orderNumberParameter / routeParameter / shipToLocParameter /
                        billToParameter / purchaseFromParameter / shipForLocParameter /
                        locNoParameter / prodServLocParameter / shipToPostParameter /
                        shipToPostISOParameter / originParameter /
                        countryProcessParameter / countryFullProcessParameter /
                        countryInitialProcessParameter / countryDisassemblyParameter /
                        originSubdivisionParameter / nhrnPZNParameter / nhrnCIPParameter /
                        nhrnCNParameter / nhrnDRNParameter / nsnParameter /
                        meatCutParameter / activePotencyParameter / catchAreaParameter /
                        fishingGearTypeParameter / prodMethodParameter /
                        refurbLotParameter / funcStatParameter / revStatParameter /
                        giaiAssemblyParameter / dimensionsParameter / cmtNoParameter /
                        ibanParameter / prodTimeParameter / versionParameter /
                        refNoParameter / couponIDNAPParameter / pointsParameter /
                        itipContentParameter / certificationRefParameter /
  
```


aquaticSpeciesParameter / opticalSensorParameter /
 paperlessCouponIDNAPParameter /
 internalParameter / mutualParameter / extensionParameter

5.10 Constructing the GS1 Digital Link URI

The following rules are derived from rules appearing in [RFC 3986] and are used for defining the general structure of a Web URI. These are particularly relevant for GS1 Digital Link URIs that are not under the id.gs1.org domain.

```

scheme                = "http" / "https"

unreserved            = ALPHA / DIGIT / "-" / "." / "_" / "~"

reserved              = gen-delims / sub-delims

pct-encoded           = "%" HEXDIG HEXDIG

gen-delims            = ":" / "/" / "?" / "#" / "[" / "]" / "@"

sub-delims            = "!" / "$" / "&" / "'" / "(" / ")" / "*" /
                        "+" / "," / ";" / "="

pchar                 = unreserved / pct-encoded / sub-delims / ":" / "@"

segment               = *pchar

reg-name              = *( unreserved / pct-encoded / sub-delims )

dec-octet             = DIGIT                ; 0-9
                      / %x31-39 DIGIT        ; 10-99
                      / "1" 2DIGIT           ; 100-199
                      / "2" %x30-34 DIGIT    ; 200-249
                      / "25" %x30-35        ; 250-255

IPv4address           = dec-octet "." dec-octet "." dec-octet "." dec-octet

IPv6address           = 6( h16 ":" ) ls32
                      / "::" 5( h16 ":" ) ls32
                      / [ h16 ] "::" 4( h16 ":" ) ls32
                      / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                      / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                      / [ *3( h16 ":" ) h16 ] "::"   h16 ":"   ls32
                      / [ *4( h16 ":" ) h16 ] "::"
                      / [ *5( h16 ":" ) h16 ] "::"
                      / [ *6( h16 ":" ) h16 ] "::"

ls32                  = ( h16 ":" h16 ) / IPv4address
                      ; least-significant 32 bits of address

h16                   = 1*4HEXDIG
                      ; 16 bits of address represented in hexadecimal

IP-literal            = "[" ( IPv6address / IPvFuture  ) "]"

IPvFuture             = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

host                  = IP-literal / IPv4address / reg-name

port                  = *DIGIT
  
```

```
hostname          = host [ ":" port ]
```

Finally, the following four rules define the syntax of a reference GS1 Digital Link URI from the concatenation of previous defined components:

```
queryStringDelim  = "&" / ";"
```

```
queryStringComp   =  
    "?" queryStringParam *( queryStringDelim queryStringParam)
```

```
uncompressedGS1webURIPattern = gslpath [queryStringComp]
```

```
referenceGS1webURI = "https://id.gs1.org" uncompressedGS1webURIPattern
```

The following rules define the syntax of a non-reference GS1 Digital Link URI from the concatenation of previous defined components. An example of usage of a non-reference GS1 Digital Link URI is when a company chooses to use their own registered Internet domain name to construct the Web URI but aligns with this specification for the format of the final part of the URI path information and query string. Note that zero or more path segments are permitted to appear after the hostname or domain name and before the start of the `gsluriPattern` defined in this specification.

```
customURISTEM     = scheme "://" hostname *( "/" segment )
```

```
uncompressedCustomGS1webURI = customURISTEM uncompressedGS1webURIPattern
```

The formal ABNF syntax for the URI should be read in combination with the GS1 General Specifications [GENSPECS] to ensure appropriate usage of Application Identifiers that represent data attributes of identified things. In particular, section 4.14 of the GS1 General Specifications [GENSPECS] provides guidance about data relationships, including invalid pairs of element strings (see section 4.14.1) and mandatory associations of element strings (see section 4.14.2). In the GS1 General Specifications [GENSPECS], section 2 specifies which identifiers are used for an application, section 3 provides definitions for each Application Identifier, while section 4 explains the management rules for each GS1 identification key.

As previously mentioned, some GS1 primary identifier keys include GS1 check digits and some also include indicator digits or extension digits that are to be used for specific purposes. Section 7 of the GS1 General Specifications [GENSPECS] provides details of AIDC validation rules and section 7.2.7 explains the GS1 check digit calculation. Nothing in this GS1 specification changes the existing validation rules that apply to the values of GS1 Application Identifiers; this document only specifies how valid GS1 AI values shall be expressed in the GS1 Digital Link structure.

Any URI that conforms to the formal syntax as defined above and that respects the relevant rules specified in the GS1 General Specifications as cited is:

1. a valid Web URI that can be dereferenced on the Web without further processing;
2. a valid expression of one or more GS1 application identifiers and their values, informationally equivalent to the same data expressed in GS1 AI syntax³

5.11 Canonical GS1 Digital Link URIs

The preceding rules provide the formal specification of GS1 Digital Link URIs in which the most commonly used identification keys can be represented as either their numeric value (their AI) or as a more developer-friendly string. As shown in section 6, the following URIs are both 'reference GS1 Digital Link URIs' and both identify the same resource:

³ There are some combinations of AIs not forbidden by the General Specifications that are not fully expressible in GS1 DL syntax. See note in section [10.1](#).

```
https://id.gs1.org/gtin/614141123452  
https://id.gs1.org/01/614141123452
```

This flexibility allows any combination of numeric and string identifiers so that:

```
https://id.gs1.org/gtin/614141123452/cpv/2A  
https://id.gs1.org/gtin/00614141123452/cpv/2A  
https://id.gs1.org/01/614141123452/22/2A  
https://id.gs1.org/01/614141123452/cpv/2A  
https://id.gs1.org/gtin/614141123452/22/2A
```

are all acceptable.

This can be extended even further since a GS1 Digital Link URI can be constructed in any domain name, may contain additional key/value pairs in the query string and so on. This flexibility is a deliberate feature of the standard to support its use in as many scenarios as possible.

However, in some contexts it is necessary to identify a *single* preferred version of the GS1 Digital Link URI. This is defined in [RFC 6596] as the *canonical URI*. Since the GS1 Digital Link URI encodes element strings as defined in the GS1 General Specifications [GENSPECS] such that

`https://example.com/gtin/614141123452` and `(01)00614141123452` are equivalent, and this document defines short string alternatives for a subset of GS1 keys, we define the canonical URI as follows:

- the domain name SHALL be `id.gs1.org`;
- convenience string equivalents for AIs SHALL NOT be used;
- GTIN-8, GTIN-12 and GTIN-13 SHALL be padded to 14 digits
- the URI query string (if present) SHALL NOT contain any other key=value pairs except for keys that are GS1 application identifiers or those defined in the Lightweight Messaging Service and Scan4Transport standards;
- key=value pairs, if present, should be sorted in lexical, not numeric, order of the key;
- for clarity, this means that the `linkType` and `context` parameters and their values are not included in the canonical GS1 Digital Link (see section [Z.7](#)).`

It follows that the canonical version of `https://example.com/gtin/614141123452/cpv/2A` is `https://id.gs1.org/01/00614141123452/22/2A`

6 Examples of GS1 Digital Link URIs

This section is informative

Please note that this chapter, as well as chapters 4 and 5, have been superseded by GS1 Digital Link Standard: URI Syntax available from <https://ref.gs1.org/standards/digital-link/uri-syntax/>. All other sections are the current version of the standard.

6.1 GTIN

```
https://id.gs1.org/gtin/614141123452  
https://id.gs1.org/01/00614141123452
```

are equivalent reference GS1 Digital Link URIs for GTIN 614141123452; the second version is canonical. They are both equivalent to the following element string:

(01)00614141123452

The following are all further valid GS1 Digital Link URIs for GTIN 614141123452 using a custom domain name e.g. example.com instead of id.gs1.org

<https://brand.example.com/gtin/614141123452>

<https://brand.example.com/some-extra/pathinfo/gtin/614141123452>

<https://retailer.example.com/some-extra/pathinfo/gtin/614141123452>

If redirection information has been specified to GS1 by the corresponding licensee of that GTIN or the GS1 Company Prefix (for GTINs constructed from GS1 Company Prefixes), a GS1 Resolver that supports GS1 Digital Link URIs will be able to effectively redirect any requests for that GS1 Digital Link URI to a corresponding URL specified by the licensee.

6.2 GTIN + CPV

<https://id.gs1.org/gtin/614141123452/cpv/2A>

<https://id.gs1.org/01/00614141123452/22/2A>

are equivalent reference GS1 Digital Link URIs for GTIN 614141123452 combined with Consumer Product Variant '2A'; the second example is canonical. They are both equivalent to the following element strings:

(01)00614141123452(22)2A

The following are further valid GS1 Digital Link URIs for GTIN 614141123452 combined with Consumer Product Variant '2A'.

<https://brand.example.com/gtin/614141123452/cpv/2A>

<https://brand.example.com/01/614141123452/22/2A>

<https://retailer.example.com/01/614141123452/22/2A>

6.3 GTIN + Batch/Lot

<https://id.gs1.org/gtin/614141123452/lot/ABC123>

<https://id.gs1.org/01/00614141123452/10/ABC123>

are equivalent reference GS1 Digital Link URIs for GTIN 614141123452 combined with Batch/Lot 'ABC123'; the second version is canonical. They are both equivalent to the following element strings:

(01)00614141123452(10)ABC123

The following are further valid GS1 Digital Link URIs for GTIN 614141123452 combined with Batch/Lot 'ABC123'

<https://brand.example.com/gtin/614141123452/lot/ABC123>

<https://brand.example.com/01/614141123452/10/ABC123>

<https://retailer.example.com/01/614141123452/10/ABC123>

6.4 GTIN + Serial Number (also known as SGTIN)

<https://id.gs1.org/gtin/614141123452/ser/12345>

<https://id.gs1.org/01/00614141123452/21/12345>

are equivalent reference GS1 Digital Link URIs for GTIN 614141123452 combined with Serial Number '12345'; the second version is canonical. They are both equivalent to the following element strings:

```
(01)00614141123452(21)12345
```

The following are further valid GS1 Digital Link URIs for GTIN 614141123452 combined with Serial Number '12345'

```
https://brand.example.com/gtin/614141123452/ser/1234
```

```
https://brand.example.com/01/614141123452/21/12345
```

```
https://retailer.example.com/01/614141123452/21/12345
```

6.5 GTIN + Batch/Lot + Serial Number + Expiry Date

```
https://id.gs1.org/gtin/614141123452/lot/ABC1/ser/12345?exp=180426
```

```
https://id.gs1.org/01/00614141123452/10/ABC1/21/12345?17=180426
```

are equivalent reference GS1 Digital Link URIs for GTIN 614141123452 combined with Batch/Lot 'ABC1' and Serial Number '12345' and with an expiry date of 26th April 2018; the second version is canonical. They are both equivalent to the following element strings:

```
(01)00614141123452(17)180426(10)ABC1(21)12345
```

The following are further valid GS1 Digital Link URIs for GTIN 614141123452 combined with Batch/Lot 'ABC1' and Serial Number '12345' and with an expiry date of 26th April 2018.

```
https://example.com/gtin/614141123452/lot/ABC1/ser/12345?exp=180426
```

```
https://example.com/01/614141123452/10/ABC1/21/12345?17=180426
```

6.6 GTIN + Net Weight

```
https://id.gs1.org/gtin/614141123452?3103=000195
```

```
https://id.gs1.org/01/00614141123452?3103=000195
```

are equivalent reference GS1 Digital Link URIs for GTIN 614141123452 combined with a net weight of 0.195 kg; the second version is canonical. They are both equivalent to the following element strings:

```
(01)00614141123452(3103)000195
```

The following are further valid GS1 Digital Link URIs for GTIN 614141123452 combined with a net weight of 0.195 kg :

```
https://example.com/gtin/614141123452?3103=000195
```

```
https://example.com/01/614141123452?3103=000195
```

6.7 SSCC

```
https://id.gs1.org/sscc/106141412345678908
```

```
https://id.gs1.org/00/106141412345678908
```

are equivalent reference GS1 Digital Link URIs for SSCC 106141412345678908; the second version is canonical. They are both equivalent to the following element string:

```
(00)106141412345678908
```

The following are further valid GS1 Digital Link URIs for SSCC 106141412345678908 :

<https://example.com/sscc/106141412345678908>

<https://example.com/00/106141412345678908>

6.8 SSCC with specified Content, Count and Batch/Lot

<https://id.gs1.org/sscc/106141412345678908?02=00614141123452&37=25&10=ABC123>

<https://id.gs1.org/00/106141412345678908?02=00614141123452&37=25&10=ABC123>

are equivalent reference GS1 Digital Link URIs for SSCC 106141412345678908 containing a count [AI (37)] of 25 instances of Content [AI (02)] 00614141123452 having Batch/Lot identifier [AI (10)] 'ABC123'; the second version is canonical. They are both equivalent to the following element strings:

(00)106141412345678908(02)00614141123452(37)25(10)ABC123

The following are further valid GS1 Digital Link URIs for SSCC 106141412345678908 containing a count [AI (37)] of 25 instances of Content [AI (02)] 00614141123452 having Batch/Lot identifier [AI (10)] 'ABC123':

<https://example.com/sscc/106141412345678908?02=00614141123452&37=25&10=ABC123>

<https://example.com/00/106141412345678908?02=00614141123452&37=25&10=ABC123>

6.9 Physical location represented by a GLN or GLN + GLN Extension

<https://id.gs1.org/gln/0614141123452>

<https://id.gs1.org/414/0614141123452>

are equivalent reference GS1 Digital Link URIs for GLN 0614141123452; the second version is canonical. They are both equivalent to the following element string:

(414)0614141123452

<https://id.gs1.org/gln/0614141123452/glnx/32a%2Fb>

<https://id.gs1.org/414/0614141123452/254/32a%2Fb>

are equivalent reference GS1 Digital Link URIs for GLN 0614141123452 combined with a GLN extension '32a/b'; the second version is canonical. Note that because the forward slash character has a special meaning within Web URIs, it is replaced with %2F, its percent encoding, when it is being used as a literal value, rather than as a URI path separator.

They are both equivalent to the following element strings:

(414)0614141123452(254)32a/b

The following are valid GS1 Digital Link URIs for GLN 0614141123452 :

<https://example.com/gln/0614141123452>

<https://example.com/414/0614141123452>

The following are further valid GS1 Digital Link URIs for GLN 0614141123452 combined with a GLN extension '32a/b' :

<https://example.com/gln/0614141123452/glnx/32a%2Fb>

<https://example.com/414/0614141123452/glnx/32a%2Fb>

7 Resolving GS1 Digital Link URIs

Please note that this chapter has been superseded by GS1-Conformant resolver Standard available from <https://ref.gs1.org/standards/resolver/>.

This section and all its subsections, except its introduction, are normative

A GS1 conformant resolver connects a GS1-identified item to one or more online resources that are directly related to it. The item may be identified at any level of granularity, and the resources may be either human or machine readable. Examples include product information pages, instruction manuals, patient leaflets and clinical data, product data, service APIs, marketing experiences and more. By adhering to a common protocol based on existing GS1 identifiers and existing Web technologies, each conformant GS1 resolver is part of a coherent yet distributed network of information resources. The remainder of this chapter defines the minimum set of requirements of a resolver that make this vision possible.

The concept of resolving an identifier through an online service is not new or unique to GS1. For example, Digital Object Identifiers, [DOI] are in common use in scientific research as identifiers for papers and datasets. The DOI for the paper that describes the discovery of the Higgs Boson is 10.1103/PhysRevD.89.032002. This can be resolved through multiple services such as <http://hdl.handle.net/> and <https://oadoi.org/> as well as the DOI organisation's own resolver at <https://doi.org/>. Add '10.1103/PhysRevD.89.032002' to any of those URL 'stubs' and you'll be redirected to the relevant paper. Other examples of existing resolver services include ORCID's for identifying researchers / authors of academic publications, and numerous identifiers for things like chemicals and drugs.

The GS1 case is similar in that GS1 keys exist independently of any online system and a GS1 conformant resolver is a service for connecting that key, including any key qualifiers, to the Web. However, it is worth highlighting an important difference between resolver services such as those used for DOIs and the GS1 case. Resolvers typically offer a one-to-one redirection, that is, resolving an identifier always leads to the same, single, resource. A GS1 conformant resolver may redirect to any number of different resources related to the given set of identifiers and there is no need for all resolvers to include the same set of possible redirects. Furthermore, although redirection is the norm, and a GS1 conformant resolver may redirect to another resolver, it may also provide content directly with no redirection.

A resolver does not need to support all primary identification keys (section 5.2) to be conformant with this standard, rather, it may support just a subset. For example, a conformant resolver may support just GTIN, or GTIN + GLN etc. However, for each supported primary identification key, all its key qualifiers and data attributes must be fully supported.

7.1 Resolver Functionality

Fundamentally, a resolver is a Web server. Therefore it SHALL operate according to the relevant standards that define HTTP 1.1 or higher for GET, HEAD and OPTIONS requests and SHALL support HTTP over TLS [HTTPS]. On top of HTTP, a resolver SHALL also implement Cross-Origin Resource Sharing (CORS) [CORS] to allow client-side Javascript Web applications to access the resolver across domains.

This specification adds additional functionalities that define a GS1 conformant resolver. These are:

1. **Decompression:** Decompress the request URI if necessary.
2. **Validation:** Validate that the URI to be resolved is conformant with this GS1 Digital Link standard and therefore contains syntactically valid GS1 elements.
3. **Providing links** and/or content:
 - a. Providing links to, or provide directly, resources related specifically to the item identified by the GS1 elements (hereafter called *identified item*)
 - b. One of the links SHALL be recognised by the resolver as the default, the resolver SHALL redirect to that URL unless there is information to the contrary. Such information may be

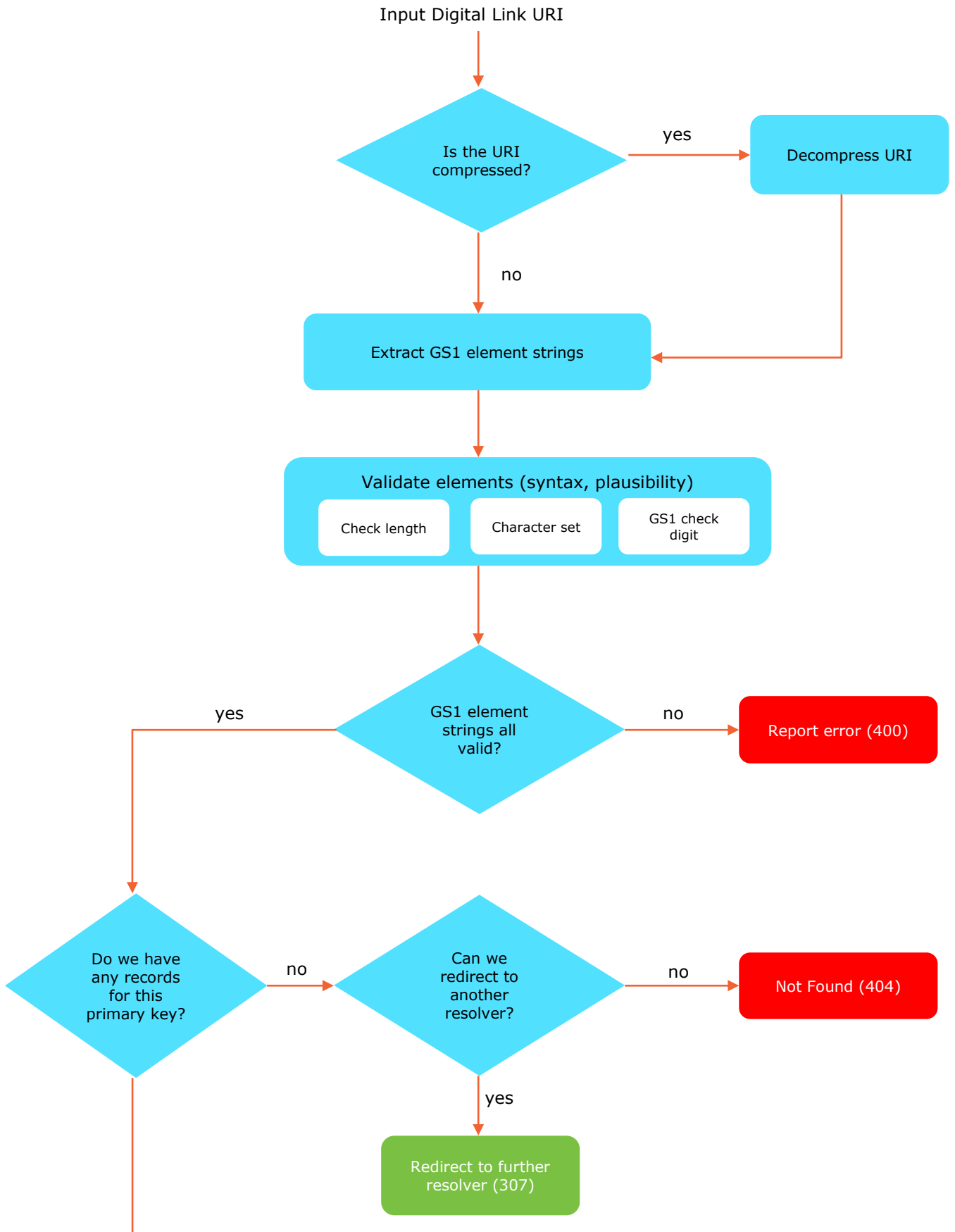
provided by setting a `linkType` parameter in the query string to the type of link desired. If a link to such a resource is known to the resolver, it SHALL redirect to that resource immediately. If the value of `linkType` is set to `all` then the resolver SHALL return a full list of links available to the client.

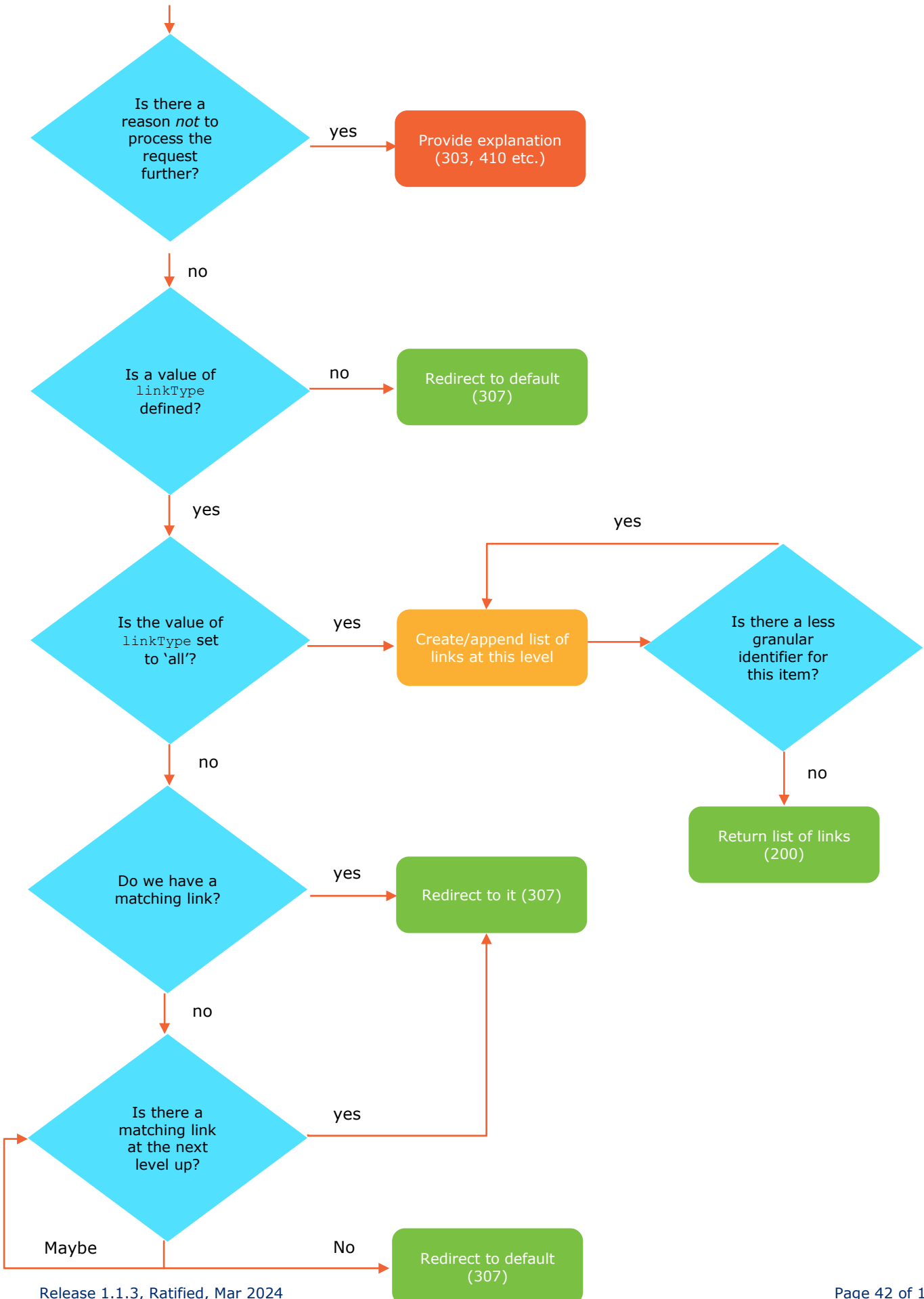
- c. Even when redirecting, a GS1 conformant resolver SHOULD expose the list of available links in the HTTP response headers [RFC8288].

Note that when redirecting, by default all key=value pairs in the query string of the original GS1 Digital Link URI to be resolved SHALL be passed on without modification. This behaviour MAY be suppressed if including the incoming query parameters cause an error at the target resource.

Each of these is expanded upon with important detail in the following sections. The flowchart below provides a visual summary.

Figure 7-1 Resolver flow chart





7.2 Decompression

As described in section 8, a GS1 Digital Link URI can be compressed to allow its direct encoding in a data carrier with limited capacity. An important feature of the compression algorithm is that it does not affect the domain name used and that the compressed GS1 Digital Link URI is still a valid URL. This means that a compressed GS1 Digital Link URI can be looked up on *exactly* the same resolver with *exactly the same result* as its uncompressed equivalent. Therefore, a GS1 conformant resolver SHALL recognise compressed GS1 Digital Link URIs and decompress them as a first processing step.

When responding to a compressed URI, a GS1 conformant resolver SHOULD NOT redirect to the equivalent uncompressed URI but SHALL expose it as follows:

1. In the Link header with a `linkType (rel)` value of `owl:sameAs`.
2. As the subject of a list of available links compiled in accordance with a `linkType=all` instruction (section 7.7.4).
3. When returning facts about the identified item expressed as RDF triples (section 9)

If redirecting to another resolver, the resolver SHOULD redirect to the uncompressed URI.

7.3 Validation

A GS1 conformant resolver SHALL validate the syntax of a GS1 Digital Link URI, as defined in this standard.

For a given request URI (decompressed if necessary) there are two basic error conditions:

1. the request URI does not contain syntactically valid GS1 identifiers;
2. the request URI is syntactically valid but the resolver does not have any information about the GS1 identifier(s).

It's important, from a developer perspective, that the resolver differentiates between these two error conditions. If the request URI is not a syntactically valid GS1 Digital Link URI, the resolver SHALL respond with an HTTP response code of 400 (Bad Request) [HTTPcodes].

If the request URI is a syntactically valid GS1 Digital Link URI, but the resolver has no information about the identified items then there are a number of possible responses:

1. redirection to an alternative resolver (with an HTTP Response code of 307 Temporary Redirect). Note that in practice a large number of redirections should be avoided as each redirection needs to be called by the client leading to a slow response time and a poor consumer experience. Implementers are encouraged to add checking routines to ensure that destination URLs do not lead to large numbers of redirects and/or circular redirections.
2. a simple 404 Not Found response.

There are other scenarios where the resolver is not able to provide any service for the identified item but does have information about *why* that is the case. For example, a licence for an identifier may have expired and not been renewed, a product is not yet on the market and no information has yet been made available. Implementations MAY use the HTTP response code 303 (See Other) to redirect to a resource that explains why the request URI cannot be resolved, at least at the time of asking. A more definite 'no longer available, please don't ask again' message can be signalled with the HTTP response code 410 (Gone). Best practice in such circumstances is to provide information about where information may be archived and how to contact the owner of the archive.

A GS1 conformant resolver SHALL NOT use a 200 OK response code with a resource that indicates an error condition.

7.3.1 Validation detail

This subsection defines the minimum validation steps that a conformant GS1 resolver SHALL perform. It does not include checking that the combination of GS1 keys, key qualifiers and attributes within the GS1 Digital Link URI conform to the General Specifications [GenSepcs] which include tables of combinations that are and are not valid. Such additional validation MAY be carried

out by resolvers. However, such validation SHALL be carried out if the resolver provides a semantic interpretation of GS1 Digital Link URIs (see section [9.3](#))

Validation should be done for each extracted value of GS1 primary identifier for which the resolver has documented support in the resolver description file as described in 8.11 and any key qualifier and/or attribute included in the request URI. The actual validation steps to be performed depend on whether the extracted value is a GS1 identification key, a key qualifier or a data attribute.

Before performing these validation tests, it is essential to convert any percent-encoded characters to their ASCII equivalent, as shown in [Table 7-1](#), because some symbol characters have a special meaning when used within Web URIs and URLs and therefore need to be 'escaped' via percent encoding (see section 2.1 of RFC 3986 [PercentEncoding]) when they are used literally.

Table 7-1 Percent-encoding of symbol characters within Web URIs

Percent-encoded	Character	ASCII code (decimal)	Name of character
%21	!	33	Exclamation mark
%20	(space)*	32	Space
%23	#	35	Octothorpe (also known as Hash or US pound symbol)
%25	%	37	Percent
%26	&	38	Ampersand
%28	(40	Left bracket
%29)	41	Right bracket
%2A	*	42	Asterisk
%2B	+	43	Plus
%2C	,	44	Comma
%2F	/	47	Solidus or Forward slash
%3A	:	58	Colon

* Note that in the query string (only), the space character may be escaped as the + character (This common practice comes from HTML forms rather than IETF's URI specifications. See <https://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.1>)

7.3.1.1 Check the length

For any value, check that its total number of digits or characters is within the limits in the ABNF grammar specified in Section [3](#) or within Figure 3.1-1 of the GS1 General Specifications [GENSPECS].

For example, if validating a GTIN value, the ABNF grammar states that the value should be exactly either 8, 12, 13 or 14 digits (`gtin-value = 8DIGIT / 12DIGIT / 13DIGIT / 14DIGIT`).

If validating a batch/lot number, the ABNF grammar states that the value should be up to 20 alphanumeric characters (`lot-value = 1*20XCHAR`) or `N2+X..20` in Figure 3.1-1 of the GS1 General Specifications [GENSPECS].

If validating a GTIN Serial number, the ABNF grammar states that the value should be up to 20 alphanumeric characters (`ser-value = 1*20XCHAR`) or `N2+X..20` in Figure 3.1-1 of the GS1 General Specifications [GENSPECS].

Similar checks should be made for other key qualifiers and data attribute values.

7.3.1.2 Check the character set

For any value, check that each character is within the character set specified in the ABNF grammar defined in section [5.1](#) or within Figure 3.1-1 of the GS1 General Specifications [GENSPECS].

For example, if validating a GTIN value, the ABNF grammar states that the value should only consist of digits (`gtin-value = 8DIGIT / 12DIGIT / 13DIGIT / 14DIGIT`) or N2+N14 in Figure 3.1-1 of the GS1 General Specifications [GENSPECS].

If validating a batch/lot number, the ABNF grammar states that the value should consist of alphanumeric characters in the 82-character subset of ISO/IEC 646 (see Figure 7.11-1 of the GS1 General Specifications [GENSPECS]) (`lot-value = 1*20XCHAR`) or N2+X..20 in Figure 3.1-1 of the GS1 General Specifications [GENSPECS].

The alphanumeric character set corresponding to Figure 7.11-1 corresponds to the following regular expression `[\x21-\x22\x25-\x39\x41-\x5A\x5F\x61-\x7A]`

The alphanumeric character set corresponding to Figure 7.11-2 corresponds to the following regular expression `[\x23\x2D\x2F\x30-\x39\x41-\x5A]`

7.3.1.3 Check the GS1 check digit (where appropriate)

For any value that corresponds to a GS1 identification key that has a GS1 check digit, perform the GS1 Check Digit calculation appropriate to that identifier as specified in section 7.9 of the GS1 General Specifications [GENSPECS] and check that it matches the value of the digit at the position where the Check Digit should appear.

Note that some GS1 identification keys do not include a check digit. Also note that the check digit is not always the final digit of a GS1 identification key. [Table 7-2](#) indicates which GS1 identification keys have a GS1 check digit - and its position within the identifier.

Table 7-2 Presence and position of GS1 check digit in various GS1 identification keys



GS1 identification key	Check Digit present?	Position of Check Digit
GTIN	Yes	Final digit of GTIN-8, GTIN-12, GTIN-13 or GTIN-14
ITIP	Yes	14 (calculated over digits 1-13)
SSCC	Yes	18 (calculated over digits 1-17)
GDTI	Yes	13 (calculated over digits 1-12)
GLN	Yes	13 (calculated over digits 1-12)
GRAI	Yes	13 (calculated over digits 1-12)
GSRN	Yes	18 (calculated over digits 1-17)
GSRN-P	Yes	18 (calculated over digits 1-17)
GSIN	Yes	17 (calculated over digits 1-16)
GCN	Yes	13 (calculated over digits 1-12)
GIAI	No	
CPID	No	
GMN	No	

Additionally, for values of the following data attributes, check digits should be checked, as shown in [Table 7-3](#):

Table 7-3 Position of GS1 check digit for various data attributes

AI for data attribute	Gen Specs data title	Position of Check Digit
02	CONTENT	Final digit of GTIN value
410	SHIP TO LOC	13 (calculated over digits 1-12)
411	BILL TO	13 (calculated over digits 1-12)
412	PURCHASE FROM	13 (calculated over digits 1-12)
413	SHIP FOR LOC	13 (calculated over digits 1-12)
414	LOC No	13 (calculated over digits 1-12)

AI for data attribute	Gen Specs data title	Position of Check Digit
415	PAY TO	13 (calculated over digits 1-12)
416	PROD/SERV LOC	13 (calculated over digits 1-12)

- 
Note: many of the validation checks on the length, character set and GS1 check digit (if any) can be performed at the time of inserting a redirection rule into the resolver for that specific identifier; the redirection will only happen for a specified identifier that is valid.
- 
Note: within some applications (such as healthcare Basic UDI-DI) the GMN has a well-defined checking scheme, however it is not considered to be part of the general GMN specification for Digital Link purposes. Note the GMN (AI 8013) has an optional check character pair that can be used in a symbol and the GMN (Basic UDI-DI) is not to be used in a symbol.

7.4 Link metadata

An important function of a resolver is to provide links to resources related to the identified item and, in many situations, immediately redirect requests to the most appropriate user experience. In order for resolvers to be able to match requests to available links, and for applications to automatically navigate those links without having to present the user with potentially confusing choices, it is necessary that each link is accompanied by standardised metadata. We draw on the Web Linking standard [RFC8288] for this purpose.

Whilst only some metadata attributes are mandatory, it is good practice to provide as many of them as possible. The attributes used in a GS1 conformant resolver are:

- The target URL itself (mandatory).
- The link relation type, i.e. the relationship between the identified item and the linked resource, such as a product information page, a warranty registration page, a related video etc. (mandatory).
- A title for the link that can be displayed to end users (mandatory).
- The media type for the content (`text/html` for HTML, `application/json` for JSON, `application/ld+json` for JSON-LD etc.).
- The human language of the resource.

Each of these is discussed in more detail below. RFC8288 defines further attributes, most notably the anchor attribute. This sets the context, that is, the subject that is described by the links. By default, the context is the source of the links which in our case is always the request URI. If the anchor attribute is set, it SHALL be set, redundantly, to the original request URI.

Other attributes defined in in RFC8288 are not discussed further in this specification.

7.4.1 The target URL

This is the link itself and is therefore required. It is the value of the `href` attribute in a hyperlink or link element. It SHOULD be the URL of a resource that is directly relevant to the identified item. It SHOULD NOT be a general page, such as a manufacturer's homepage or a set of search results.

7.4.2 The link relation type (linkType)

Links SHALL be annotated with a single `rel` attribute that contains a whitespace separated list of the kind(s) of resource available by following the link. The relationship(s) that are the value of the `rel` attribute SHOULD be defined either in the GS1 Web vocabulary [GS1Voc] or the schema.org vocabulary that is valid for the kind of identified item (usually both a `gs1:Product` and a `schema:Product`) (see section 7.9). If neither of those are suitable, then you MAY use your own URI which should dereference to information about the relationship. Using this method is likely to greatly reduce interoperability, however. It is therefore much better to seek to add terms to the GS1 or schema.org vocabularies.


For example, resolving the GTIN of a product might return links with `rel` values such as

- `schema:isAccessoryOrSparePartFor` (with the GS1 Digital Link URI for the product for which the request URI identifies an accessory or spare part as the value of the target URL)
- `gs1:hasAllergenInfo` (with the URL of the allergen information as the value of the target URL)

If the target URL points to a resource that fulfils a number of roles, such as a product information page that includes nutritional and allergen information, then the value of the `rel` attribute would be `gs1:pip gs1:nutritionalInfo gs1:allergenInfo`.

The use of compact URIs (CURIEs), i.e. the `schema:` and `gs1:` prefixes, is discussed further in section [7.11](#).

It is not an error for resolvers to offer multiple links of the same type, even if the linked resources are in the same (human) language or machine readable representation, see section [7.7.5](#) for how these can be disambiguated.

 **Note:** elsewhere in this document we use 'linkType' as a shorthand for link relation type not the media type as described below.

7.4.3 A title for the link

The link's `title`, which is mandatory, is provided for use in applications that present links as options for an end user to follow. The link's title SHOULD be in the same language as the destination resource. Where the target resource is multilingual, consider using `title*` as described in [RFC8187]. Link types defined by GS1 include a default title but this can be overridden.

7.4.4 The Media Type for the content

The `type` attribute provides a hint about the likely format (the serialisation) of the linked resource. Media Types are used for this, such as `text/html`, `application/json`, `application/ld+json` etc. The normative list of Media Types is maintained by IANA [IANA MT].

As noted, the term 'linkType' used elsewhere in this document refers to the link relationship type, the `rel` attribute, not the media type.

7.4.5 The human language of the linked resource

For human-readable resources, such as Web pages, or data that contains text strings in a specific language, it's a good idea to hint at the language used in that resource. This is achieved using the `hreflang` attribute, the value for which should be provided in accordance with IETF Best Current Practice for identifying languages [BCP47]. RFC8288, which defines the HTTP Link Header, supports the provision of multiple `hreflang` attributes, indicating that the linked resource contains, or through language negotiation is available in, multiple languages. However, this is not true for all representations of links, notably HTML (see section [7.7.4](#)).

7.5 Provision of links in HTTP headers

All resolver responses, including redirection, SHOULD include available links for a given request URI in the HTTP Link header.

Example

Imagine that the resolver has two links for a given identified item as shown in the table below.

Table 7-4 Examples of values for the different attributes of a link

	Link 1	Link 2
<code>href</code>	<code>https://example.com/product/ingredients</code>	<code>https://example.com/product/pip</code>
<code>rel</code>	<code>gs1:ingredientsInfo</code>	<code>gs1:pip gs1:instructions</code>

	Link 1	Link 2
title	Ingredients (Ingrédients)	Manufacturer's description
type	application/ld+json	text/html
hreflang	en, fr	en

These links would be represented within an HTTP Link Header are as follows:

```
Link: <https://example.com/product/ingredients>;
rel="https://gsl.org/voc/ingredientsInfo"; title="Ingredients (Ingrédients)";
type="application/ld+json"; hreflang="en"; hreflang="fr", <https://example.com/pip>;
rel="https://gsl.org/voc/pip https://gsl.org/voc/instructions";
title="Manufacturer's description"; type="text/html"; hreflang="en"
```

The provision of links in other formats is discussed in section [7.7.4](#).

7.6 Associating links with identified items

Links can be associated with identified items at any level of granularity. For example, an identified item might have links defined as follows:

GTIN

- Product information (default).
- Instruction manual (applies to all trade items with this GTIN).

GTIN + Batch/Lot

- Traceability information (specific to this batch of this item).

GTIN + CPV

- Recycling information (perhaps specific to seasonal packaging).

GTIN + serial number

- Registration information (specific to this individual item).

This is one of the key reasons for GS1 Digital Link URIs being structured in the way they are, with identifiers becoming progressively more specific from left to right, separated by slashes, as distinct from attributes that can appear in any order in the query string. In the following section on Redirection, this becomes particularly important when selecting the appropriate link(s) for a given item. To follow the example above, if the request URI is at the CPV level and asks for the instruction manual, the resolver 'works up the tree' until it finds the appropriate link at the GTIN level.

A resolver MAY support grouping of top level identifiers. That is, a resolver may implement a system through which a request for links related to one primary identification key can be serviced by links defined for another. This has potential use when providing information related to products sold in different territories with trivial differences, such as the patient information leaflet for the same drug but in different languages, or an electronics item sold with a different mains lead for different countries.

The precise mechanism for doing this is not defined as it is likely to be driven by commercial decisions that will have no effect on the technological solution employed.



Note: Great care must be taken when doing this to ensure that information provided by following a given `linkType` about every identified item in the group applies to all.

7.7 Redirection

The expectation is that in most cases, a resolver will respond to a request by redirecting to an appropriate resource. Provision of content directly (section [7.8](#)) is possible but it is anticipated that this will be the exception rather than the norm. This section defines how resolvers should perform

redirection and how GS1 key licensees and client applications can exercise control over the redirection process.

As noted in section [7.1](#), even when redirecting, conformant resolvers SHOULD expose all links available to the client in the HTTP Link header. This is primarily to provide all client applications the ability to discover the links available to it through a HEAD request or by interrupting the redirect. Even where only one link is available, the `linkType` and other link metadata will be exposed as that is of potential value to client applications.

7.7.1 Pattern-based redirection

Resolvers MAY support redirection of all request URIs that match a given pattern to another resolver based on a template. In the GS1 context, this might be at the GS1 Member Organisation level or the GS1 Company Prefix level and amounts to a simple passing of the request on to another service without further processing. When exposing the target URL in the Link header, the `linkType` (the `rel` value) SHALL be `gs1:handledBy`, indicating that the request URI is handled by the target resolver.

It is anticipated that further work will be done to define pattern-based redirection for sets of GS1 Digital Link URIs. This is likely to be in the broader context of a defined ingestion API for GS1 conformant resolvers as part of a future iteration of this standard.

7.7.2 Default linkType and default link (resolver side)

GS1 key licensees are encouraged to provide multiple links for each item to serve different end users. This is a key feature of the GS1 Digital Link ecosystem that underpins the notion of *one barcode performing multiple functions*. However, it is highly likely that the end user will expect to receive content (either machine or human readable) rather than a list of links. Likewise, key licensees will be keen to provide end users with the information they're most likely to find useful. On the other hand, developers need to be able to discover and serve what they deem to be the content that best suits the needs of their users. These needs lead to the following requirements.

For each identified item there SHALL be a default `linkType`

As noted in section [7.1](#) resolvers MAY support the definition of multiple links of the same `linkType`. These MAY be distinguished by human language, media type and more. Where this happens, exactly one link SHALL be designated as the default for that `linkType`.

This means that:

1. There is always a default `linkType`.
2. Within that `linkType`, there is always a default link.

Resolvers SHALL redirect to the default link *unless* there is information in the HTTP request that overrides this.

7.7.2.1 Granular identifiers

As discussed elsewhere in this standard, the structure of a GS1 Digital Link URI is designed such that the identification becomes more granular as you work from left to right in the path information, i.e. there's a subclass relationship (section [9.6](#)). So if the URI includes a GLN and a GLN extension, any facts at the GLN level also apply at the GLN extension level. Likewise with an SGTIN, any fact about the GTIN also applies to the individual instance identified by the serial number. The reverse is not true: facts about the GLN extension may not apply to the GLN, facts about the serialised item may not apply to the GTIN. As an analogy, mammals are a subclass of animal, that is, all mammals are animals but not all animals are mammals.

This means that if the request URI identifies the item at a fine level of granularity and no suitable `linkType` is available at that fine level level, the resolver SHALL inspect the links available at the higher level(s) (working from right to left in the URI path information) to look for a match there. As a consequence, it is perfectly permissible to set default links at the highest level and not further down the tree, if appropriate for the operating environment of the resolver. What is important is that for any entry point, that is, for any GS1 Digital Link URI, no matter how granular, there SHALL be a default `linkType` available either at the entry level or at a higher level.

This has positive consequences for resolvers that support a subset of the GS1 identifier space.

As noted in section [7.3.1](#), resolvers MAY support a subset of GS1 primary keys, rather than all of them. For example, it may support only GTIN and GLN, or GLN, SSCC and GRAI. The list of supported primary keys SHOULD be declared in the Resolver Description File (section [7.11](#)). For each supported primary key, a resolver SHALL support all its key qualifiers. So if the resolver supports GTIN, it must also be able to process GS1 Digital Link URIs that include CPV, batch/lot and serial number; support for GLN entails support for GLN extensions and so on. However, this does not mean that such a resolver must also support the provision of links at granular levels of identity since they can simply be provided at the primary key level and still be conformant.

7.7.2.2 Examples

Some examples should help to clarify this situation.

Example 1: An FMCG manufacturer operates their own resolver. For all identified items in the resolver's database there is exactly one link which is to the product information page. In this situation all request URIs result in a redirection to the product information page with the same `linkType` of `gs1:pip`.

Example 2: A GS1 Member Organisation operates a resolver. A consumer electronics manufacturer sets up links for one of their products as shown in the table below.

<code>linkType</code>	Resource description
<code>gs1:pip</code> (default)	Product information page
<code>gs1:instructions</code>	The instruction manual
<code>gs1:certification</code>	A document showing compliance with the relevant regulations

This is the usual situation, but for a limited period, the manufacturer wants to point potential consumers to a promotional campaign and adds another link so that their full set is:

<code>linkType</code>	Resource description
<code>gs1:pip</code>	Product information page
<code>gs1:instructions</code>	The instruction manual
<code>gs1:certification</code>	A document showing compliance with the relevant regulations
<code>gs1:promotion</code> (default)	Entry point for a promotional campaign

Note that the default `linkType` is now `gs1:promotion`.

Example 3: An independent organisation operates a repository of electronic patient information leaflets on behalf of a variety of pharmaceutical manufacturers. For each drug, they offer two types of information: one designed for patients (`gs1:epil`) and one for clinical staff (`gs1:smpc`). The leaflets are available in multiple languages and so there are multiple links for both supported link types. The manufacturer provides the list of available links as shown below.

<code>linkType</code>	Resource description
<code>gs1:epil</code> (default)	Patient information (Dutch) (default)
	Patient information (French)

linkType	Resource description
	Patient information (German)
gs1:smpc	Specific product characteristics (Dutch) (default)
	Specific product characteristics (French)
	Specific product characteristics (German)

Notice in particular the two levels of default: the default `linkType` and, within each `linkType`, the default link. In the absence of any information to the contrary, this resolver would serve the Dutch language variant of the patient information leaflet, but all the other options are available to client applications as described in the subsections below.

Example 4: A power tool manufacturer operates its own resolver and encourages its customers to register their purchase by including a GS1 Digital Link URI on the packaging, as follows:

```
https://tools.example.com/gtin/614141123452/ser/12345
```

The manufacturer sets the default `linkType` to `gs1:instructions` which points to a video showing how the tool can be used. This is set at the GTIN level and so any consumer scanning the tool's packaging in any store would see the same video. However, the same GS1 Digital Link URI can be scanned using the retailer's app which asks for a link type of `gs1:registerProduct`. Combining information from both the retailer app, which includes information about the customer, with the manufacturer's information about the serialised product makes the registration process trivially easy for the consumer.

7.7.3 Requesting a specific link type (client side)

Applications may wish to provide their users with a specific type of content for each item, irrespective of the default set by the key licensee. For example, a dietary application may want to always follow a link to nutritional information, or a clinical application may always want to follow a link to a patient information leaflet.

Applications MAY include a request for a specific type of link by including a `linkType` parameter in the query string with the value of the desired link type. For example:

```
linkType=gs1:nutritionalInfo
```

```
linkType=gs1:epil
```

- If the specified `linkType` is available, the resolver SHALL redirect to the default for that `linkType` *unless* the client request includes information that leads to a better match (see sections [7.7.5](#) and [7.7.6](#)), in which case, it SHALL redirect to that alternative.
- If the specified `linkType` is not available the resolver SHALL redirect to the default link of the default `linkType` for the item.

7.7.4 Requesting all available links

The client request the full list of links available to it by setting the value of the `linkType` parameter to `all`. This amounts to a request by the client for a list of the options available to it, rather than a request for information about the identified item. In this situation, the resolver SHALL NOT redirect the request but, instead, return the set of available links. If the resolver provides content directly without any redirection (section [7.8](#)), the returned list SHALL be a single item that is the original request URI.

If the request GS1 Digital Link URI includes one or more key qualifiers, any links associated with each level up to the primary key SHALL be included.

This specification does not limit the techniques that may be used to represent the links associated with an identified item. It does, however, specify a minimum. As noted, for all responses, the resolver SHALL provide the list of available links in an HTTP Link Header. When returning the list of

available links as a resource, a GS1 conformant resolver SHALL return a JSON array containing the link objects. Via content negotiation [RFC7231] it SHOULD also provide JSON-LD and MAY also provide an HTML document

In the absence of an `Accept` header, it is up to the resolver to decide which representation is returned but it is suggested that the human readable response is likely to be the better choice.

The links in Table 8-4 can be represented as a JSON object thus:

```
[ { "href"      : "https://example.com/product/ingredients",
    "anchor"    : "https://example.com/gtin/614141123452",
    "rel"       : [ "https://gs1.org/voc/ingredientsInfo" ],
    "title"    : "Ingredients (Ingrédients)",
    "type"     : "application/ld+json",
    "hreflang" : [ "en" , "fr" ] },
  { "href"      : "https://example.com/product/pip",
    "anchor"    : "https://example.com/gtin/614141123452",
    "rel"       : [ "https://gs1.org/voc/pip",
                  "https://gs1.org/voc/instructions" ],
    "title"    : "Manufacturer's description",
    "type"     : "text/html",
    "hreflang" : [ "en" ] },
  { "href"      : "https://example.com/gtin/614141123452.html",
    "anchor"    : "https://example.com/gtin/614141123452",
    "rel"       : [ "alternate" ],
    "title"    : "Available links as a Web page",
    "type"     : "text/html",
    "hreflang" : [ "en" , "fr" ] }
]
```

This JSON follows the format proposed in an internet draft on the IETF standards track [Linkset], however, alternative serialisations are acceptable (note section [10.1](#)). Note that since this JSON object exists in its own right and may be processed in any context, the original GS1 Digital Link URI for which these links are available needs to be stated explicitly as the value of the `anchor` property. The value `https://example.com/gtin/614141123452` is used as the anchor in this example.

It is good practice to cross link all available formats for machine discovery. This is shown as a third link in the preceding JSON example which links to the HTML representation below. This in turn links back to the JSON version through an HTML `<link />` element (the `alternate` keyword is defined in the IANA link relation types registry).

The same links within a HTML page:

```
<a href="https://example.com/product/ingredients"
    rel="https://gs1.org/voc/ingredients"
    title="Ingredients (Ingrédients)"
    type="application/ld+json"
    hreflang="en">Ingredients (Ingrédients)</a>

<a href="https://example.com/product/pip"
    rel="https://gs1.org/voc/pip
        https://gs1.org/voc/instructions"
```

```

    title="Manufacturer's description"
    type="text/html"
    hreflang="en">Manufacturer's description</a>
<link href="{requestURI}.json"
    rel="alternate"
    title="Available links as a JSON object"
    type="application/json"
    hreflang="en" />
  
```

Note that HTML has no way to indicate that the linked resource is available in multiple languages since the `hreflang` attribute cannot be repeated and the value must be a single language tag.

The same links might be provided as JSON-LD (i.e. a serialisation of RDF) thus:

```

{
  "@context": {
    "dcterms": "http://purl.org/dc/terms/",
    "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#",
    "xsd": "http://www.w3.org/2001/XMLSchema#"
  },
  "@graph": [
    {
      "@id": "https://example.com/gtin/614141123452",
      "gs1:ingredientsInfo": {
        "@id": "https://example.com/product/ingredients"
      },
      "gs1:instructions": {
        "@id": "https://example.com/product/pip"
      },
      "gs1:pip": {
        "@id": "https://example.com/product/pip"
      }
    },
    {
      "@id": "https://example.com/product/pip",
    }
  ]
}
  
```

```
"dcterms:format": "text/html",
"dcterms:title": {
  "@language": "en",
  "@value": "Manufacturer's description"
},
{
  "@id": "https://example.com/product/ingredients",
  "dcterms:format": "application/ld+json",
  "dcterms:title": [
    {
      "@language": "en",
      "@value": "Ingredients"
    },
    {
      "@language": "fr",
      "@value": "Ingrédients"
    }
  ]
}
]
```

For ease of reading, the above JSON-LD can be serialised as Turtle thus:

```
@prefix dcterms: <http://purl.org/dc/terms/>.
```

```
@prefix gs1: <https://gs1.org/voc/>.
```

```
<https://example.com/gtin/614141123452>
  gs1:ingredientsInfo <https://example.com/product/ingredients>;
  gs1:pip <https://example.com/product/pip>;
  gs1:instructions <https://example.com/product/pip>.
```

```
<https://example.com/product/ingredients>
  dcterms:title "Ingredients"@en;
  dcterms:title "Ingrédients"@fr;
  dcterms:format "application/ld+json".
```

```
<https://example.com/product/pip>
  dcterms:title "Manufacturer's description"@en;
  dcterms:format "text/html".
```

7.7.5 The context keyword

As noted in section [7.4.2](#), it is possible to define multiple links of the same `linkType` for an identified item. The `context` parameter (in the query string) MAY be used to disambiguate between multiple links that match the given `linkType`. Examples:

- A food product might link to multiple recipe pages and the `context` parameter might distinguish between different sub categories like vegetarian, vegan, gluten-free etc.
- Information about a product may vary slightly between different territories.
- A link to availability and price information might use the `context` parameter to distinguish between different retailers.
- The GS1 Lightweight Messaging Standard for Verification of Product Identifiers [LMS] provides a concrete example: it envisages that when using the particular `linkType` value of `verificationService`, the `context` parameter can be set to a specific value such as `dscsaSaleableReturn` to indicate that a verification service for product identifiers should be configured to use a profile defined by the US Drug Supply Chain Security Act; Saleable Returns, whereas a different value of the `context` parameter would cause the target resource to use a different profile, e.g. for a different jurisdiction or regulatory regime.

The value space of the `context` parameter is not defined in this specification and resolvers are not obliged to support it. For resolvers that do recognise it, the resolver description file (section [7.11](#)) SHOULD declare this and provide values that it recognises in one or other of the following ways:

- a. by enumerating recognised values;
- b. by linking to an external list of values.

Examples of this are included in section [7.11](#).

Whether the resolver processes the `context` parameter or not, as with all query string parameters, it SHALL be passed on unchanged in any redirection for potential processing by the target resource, except as described in section [7.7.8](#).

7.7.6 Recognising the user's language and requested media type

As with any Web server, GS1 conformant resolvers MAY process the client's HTTP `Accept` and `Accept-Language` headers, if present, and return links accordingly. This means that a single request URI may cause a redirection to different target URLs depending on the values of the headers.

`Accept` and `Accept-Language` headers are hints to the resolver and the precise method of how they should be processed is not defined. Apache defines a widely used algorithm that can be used to process the headers [Apache-conneg] but even this is not fully deterministic. The behaviour set out in section [7.7.2](#) is designed to overcome this as there is always a default link to follow if there's no precise match available.

7.7.7 Default linkType set to all

Resolvers MAY support the setting of the default `linkType` to `all`. This can be applied at any level. What this means is that where a specific `linkType` is not requested, or a request for a specific `linkType` cannot be fulfilled, especially taking language and media type into account, the resolver will return the full list of links available to the client application.

Since support for this function is optional, if implemented it SHOULD be declared in the resolver description file (section [7.11](#)).

Furthermore, if returning a list of available links *because the requested item is not available*, the resolver SHALL NOT use a 200 OK response code. Instead it SHALL use either of the following:

- 406 (*Not Acceptable*)
- 300 (*Multiple Choices*) noting that, according to the specification of this header “If the server has a preferred choice of representation, it SHOULD include the specific URI for that representation in the Location field; user agents MAY use the Location field value for automatic redirection. This response is cacheable unless indicated otherwise.”

7.7.8 Handling the query string

If present, the query string in an uncompressed GS1 Digital Link URI carries attributes of the identified item and does not form part of the GS1 standard identifier itself. Implementations MAY make use of query string parameters but this specification does not define a particular behaviour, except for their semantic interpretation (section 9.7), `linkType` and `context`. As a consequence, it's important that query strings are normally preserved through any redirection.

When redirecting, by default, a resolver SHALL transmit the entirety of the query string in the request URI to the target destination. This behaviour MAY be suppressed, i.e. the query string parameters not passed, if the target resource is adversely affected. The reason for this is as follows:

- Resolvers are not required to process the query string, therefore resolvers may not understand the query string
- The target resource might understand the query string therefore the resolver should simply pass everything through.
- This is usually harmless, even where the target resource doesn't understand the query string parameters since Web applications typically ignore anything they don't understand. However, some Web resources are less tolerant and therefore can be adversely affected by the presence of key=value pairs they don't understand.

7.8 Providing content directly

Operators of resolver services may provide content and user experiences directly from a GS1 Digital Link URI. The nature of that content is, of course, entirely up to the operator. However; best practice recommendation is that content should be relevant to the specific item identified and not, for example, to a manufacturer's homepage or multiple products from the same brand.

A resolver can be part of any website. Today, the websites of many retailers or manufacturers use page URLs based on internal Stock-Keeping Units (SKUs) rather than global GS1 identifiers such as the GTIN. Such websites might implement resolver functionality in order to provide aliasing or internal redirection within the website, so that the page for a specific product is available through its existing URL based on an internal SKU but can also be accessed by using the GS1 Digital Link URI syntax using that domain name. Such an internal resolver within a website has an internal record of the mapping between each GTIN and the internal SKU and uses this information to perform internal redirects. Note that when a resolver operates internally within a website, it does not need to formally issue any redirection responses. It can simply serve the corresponding page or resource without requiring the client to make a further request.

7.9 Supported link relation types

The Web Linking standard [RFC8288] defines a registry of general purpose link relation types [IANA LR] such as `alternate`, `describedby` and `icon`. These can be used without qualification in HTTP Link headers. They are similar to, but managed and defined separately, from the list of link types that can be used in HTML documents [HTML LT]. Neither list includes the detailed set of link relation types needed for GS1 conformant resolvers, which are unlikely to be of sufficiently broad applicability to warrant addition to those registries. Therefore, the GS1 Web vocabulary [GS1Voc] defines a set of link relation types (`linkTypes`). The Web Vocabulary is managed separately under the GS1 Standards Management Process but this standard includes a specific method for future change management of link types.

schema.org, the vocabulary maintained by members of search engine companies and the wider Web community, offers a small number of properties that are suitable for use as link types in a resolver directly, however, others are defined for a much broader information space than envisaged for GS1 conformant resolvers. Where relevant, the Web vocabulary defines relationships between GS1 link types and schema.org terms using semantic relationships defined in SKOS [SKOS].

The extension mechanism defined in the Web Linking standard states that values used as a link relation type other than those listed in the IANA registry must be a URI but may be serialised as strings that can be converted to a URI. We take advantage of this and express link types as CURIEs [CURIE] that is, compact URIs.

Referring to the namespaces listed in section 3, for example, `gs1:pip` expands to `https://gs1.org/voc/pip`.

GS1 conformant resolvers that support multiple links per item SHALL recognise the link types defined in the `gs1:` namespace and SHOULD recognise relevant properties in the `schema:` namespace. GS1 conformant resolvers MAY recognise further namespaces in which case they and the recognised prefixes SHOULD be declared in the resolver description file (see section 7.11).

Table 7-5 shows some example link relation types defined in the Web vocabulary at the time of publication. In the case of any variance between these examples and the definitions published in the Web vocabulary, the latter is normative. The word 'document' used in the definitions means any kind of document: human or machine readable, text, image, video etc. The type of document is given in the relevant HTTP `content` header (as a media type) and/or in the document itself.

Table 7-5 Examples of the initial set of link relation types for use in conformant GS1 resolvers. Further link types MAY be added to the GS1 Web vocabulary as described in section 7.10.

Link type	Definition	Default English title
<code>gs1:pip</code>	The URL of a document that provides information about the identified item, typically operated by the brand owner or a retailer of the product. It may include links to further information, product description, specifications etc. N.B. The page may be human or machine readable, or a combination of the two (such as an HTML page with embedded structured data).	Product information page
<code>gs1:quickStartGuide</code>	A document, video or graphic that shows the key features needed to be understood to begin using the item.	Quick start guide
<code>gs1:allergenInfo</code>	A document describing the allergens in the product.	Allergen information
<code>gs1:whatsInTheBox</code>	A document describing all the individual items in a packaged item	What's in the box
<code>gs1:certificationInfo</code>	Information on certification to which the product complies.	Certification information
<code>gs1:traceability</code>	A link to traceability information about the product	Traceability information
<code>gs1:recallStatus</code>	A link to information about whether the product has been recalled or not, typically an API	Recall status

Link type	Definition	Default English title
gs1:recipeInfo	A website containing recipes associated with the product.	Recipe website
gs1:epil	Link to an electronic patient information leaflet	Patient information
gs1:smpc	Link to Summary Product Characteristics for healthcare professionals	Summary product characteristics (SmPC)
gs1:registerProduct	A link to an entry point for registering ownership of a product including for warranty purposes	Register purchase
gs1:socialMedia	A link to a social media channel. The title will typically be replaced by the name of the channel.	Social media
gs1:support	A link to a source of support such as a helpdesk, chat support, email etc.	Support
gs1:purchaseSuppliesOrAccessories	A link to a page where supplies or accessories for the item can be purchased or ordered	Purchase supplies or accessories
gs1:hasRetailers	A link to a list of retailers for this item	Retailers

7.10 Link type maintenance

Requests for new terms to be added may be made through a GSMP Work Request and, when considering the request, the following rules will be applied:

- Link relation types must be as broad as possible while conveying a specific meaning. For example, `gs1:allergenInfo` links to a document that provides allergen information in some way and is broad enough to encompass a variety of cases. Requests for new terms that point to specific kinds of allergen are unlikely to be accepted, noting that any descriptive title can be applied to a link (i.e. the default can be overridden). This rule is designed to keep the number of link relation types as low as possible while still covering the full range of use cases.
- Link relation types SHALL NOT be deleted from the GS1 Web vocabulary, they MAY, however, be deprecated and therefore their future use be discouraged.
- Definitions MAY be clarified but not substantially altered. If necessary, deprecate the existing link relation type and define a new one.
- GS1 link types are defined using lower camel case (e.g. `gs1:nutritionalInfo`) using formal semantics as part of the broader GS1 Web vocabulary. This follows best practice for defining vocabulary terms in this manner (using RDF). However, the Web Linking standard [RFC8288] requires that link relation types are all lower case. Therefore link types should be regarded as case insensitive.
- Link relation types SHALL NOT use the same terms as those defined in other GS1 standards, notably the Global Data Dictionary [GDD], unless the meaning is identical.

7.11 Resolver description file

This standard does not define a single resolver; rather, it defines the concept of a GS1 conformant resolver. Different resolvers are likely to serve different needs and there is no requirement, nor any expectation, that every resolver will offer the same set of links or content for the same identified item. Equally, it is not expected that every resolver will offer support for every combination of GS1 key or qualifier. In other words, it is the *behaviour* of the resolver that is standardised, not the specific service(s) that it links to or provides directly.

This, coupled with the support for extension key=value pairs in the query string (section [5.9.1](#)), means that it is possible for resolvers to offer different capabilities. A resolver might, for example, support a specific extension parameter or link types defined in an additional namespace other than schema.org and the GS1 Web Vocabulary. It may include additional, proprietary compression mechanisms that sit on top of the standard GS1 approach see flowchart C14 in section [8.8](#).

Finally, note that the GS1 Digital Link URI syntax allows additional path elements between the domain name and the primary key, so that `https://example.com/extra/path/01/614141123452` is valid.

Such flexibility is deliberate and allows different businesses to use GS1 Digital Link in the way that best suits them with the minimum of restriction. On the downside, the flexibility reduces interoperability between resolvers. To address this, we define the Resolver Description File. It provides a machine-readable description of the resolver's capabilities using the terms defined below.

1. A name for the resolver
 - a. property: name
 - b. type: string
2. The resolver root (the `customURIstem` as defined in section [5.10](#)) (**required**)
 - a. property: resolverRoot
 - b. type: URI
3. The namespace(s) of supported link types and associated prefixes (see section [7.9](#)). If this is omitted, support for both the schema.org and GS1 Web vocabulary namespaces is assumed
 - a. property: supportedLinkType
 - b. type: array of objects
 - i. property: namespace
 - ii. type: URI
 - iii. property: prefix
 - iv. type: string
3. Whether `all` is a supported default `linkType` (section [7.7.7](#))
 - a. property: linkTypeDefaultCanBeAll
 - b. type: Boolean
 - c. default: false
4. Enumerate which Primary Keys & related Qualifiers are supported by the resolver (**required**)
 - a. property: supportedPrimaryKeys
 - i. type: array
 - ii. Allowed values: "all", "01", "8006", "8013", "8010", "410", "411", "412", "413", "414", "415", "416", "8017", "8018", "255", "00", "253", "401", "402", "8003", "8004"
5. The supported values (if any) for the context keyword. These can be provided through simple enumeration or by linking to an externally managed list.
 - a. property: supportedContextValuesEnumerated
 - b. type: array of strings

- c. property: supportedContextValuesExternal
 - d. type: URI
6. Whether or not the resolver supports language variants so that redirects (or content) may vary according to a client's language (section [7.7.6](#)).
 - a. property: supportsLanguageVariants
 - b. type: Boolean
 - c. default: true
7. Whether the resolver provides a semantic interpretation of request URIs
 - a. property: supportsSemanticInterpretation
 - b. type: Boolean
 - c. default: false
8. Whether the resolver validates the combination of AIs according to the GS1 General Specifications
 - a. property: validatesAIcombinations
 - b. type: Boolean
 - c. default: false
9. Contact details for the resolver operator;
 - a. property: contact
 - b. type: VCard
10. A pointer to a document that describes any extensions the resolver supports, such as any extended key=value pairs and/or an additional supported compression/decompression capability
 - a. property: extension profile
 - b. type: URI
11. The URL of a complete set of GS1 Digital Link URIs and their associated links available from this resolver
 - a. property: dataDump
 - b. type: URI

Providing this information allows applications to automatically detect a resolver's capabilities and act accordingly. Furthermore, it explicitly asserts the sovereignty of URLs within its own domain name. This is important for the architecture of the World Wide Web and is discussed further in section [9.3](#). For this reason, a GS1 conformant resolver SHALL make a resolver description file available at `/.well-known/gs1resolver`, in line with Defining Well-Known Uniform Resource Identifiers (URIs) [RFC8615]

7.11.1 JSON schema

The JSON schema against which resolver description files may be validated is published at <https://id.gs1.org/resolverdescriptionfile.schema.json>.

7.11.2 Example

As an example, the GS1 Global Office resolver's description file (available from <https://id.gs1.org/.well-known/gs1resolver>) is reproduced below.

```
{
  "name": "The GS1 Global Office resolver",
  "resolverRoot": "https://id.gs1.org",
```

```

    "supportedLinkType" : [{"namespace":"http://gs1.org/voc/",
"prefix":"gs1:"}, {"namespace":"http://schema.org/", "prefix":"schema:"}],
    "supportedPrimaryKeys": ["all"],
    "supportedContextValuesEnumerated": ["dscsaSaleableReturn"],
    "supportedContextValuesExternal": [{"nameOfList":"ISO-3166 Alpha-2",
"url":"https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes"}],
    "linkTypeDefaultCanBeAll": true,
    "contact": {"fn":"GS1 AISBL", "hasAddress":{"streetAddress":"Avenue Louise 326",
"locality":"Brussels", "country-name":"Belgium", "postal-code":"1050"},
"hasTelepone":"tel:+32 2 788 78 00"},
    "supportsLanguageVariants":true,
    "supportsSemanticInterpretation": true,
    "validatesAIcombinations": true
  }

```

7.12 Conformance statement

Summarising the preceding discussion, a GS1 conformant resolver:

1. SHALL support HTTP 1.1 (or higher) GET, HEAD and OPTIONS requests.
4. SHALL support HTTP Over TLS (HTTPS)
5. SHALL support CORS
6. SHALL be able to decompress a URI to generate a GS1 Digital Link URI, in accordance with section [8](#).
7. If handling a compressed request URI, it SHALL expose the uncompressed URI in the Link response header with a `rel` value of `owl:sameAs`.
8. MAY support additional decompression algorithms (flowchart C4 in section [8.8](#)).
9. SHALL extract and syntactically validate the URI in accordance with sections [3](#) and [7.3.1](#), and report errors with an HTTP response code of 400.
10. A GS1 conformant resolver SHALL NOT use a 200 OK response code with a resource that indicates an error condition.
11. MAY support additional key-value pairs in the query string (section [5.9.1](#))
12. SHALL respond to a query parameter of `linkType` set to `all` by returning a list of links available to the client application. The list SHALL be available as JSON, SHOULD be available as JSON-LD and MAY be available in HTML and any other formats, served through content negotiation. If the request GS1 Digital Link URI includes one or more key qualifiers, any links associated with each level up to the primary key SHALL be included (section [7.7.4](#)). The URI used as the subject of facts presented SHALL be the uncompressed version.
13. SHOULD expose the full list of links available to the client in an HTTP Link header when redirecting.
14. SHALL recognise one available `linkType` as the default for any given request URI and, within that, SHALL recognise one default link which may be at a less granular level than the request URI (section [7.7.2](#)).
15. All links exposed SHALL include the target URL, the link relationship type (the `linkType`) and a human-readable title (section [7.4](#))
16. SHALL redirect to the requested `linkType` if available (section [7.7.3](#)).
17. By default, SHALL pass on all key=value pairs in the query string of the request URI (if present) when redirecting.

18. MAY support use of the Accept-Language and Accept HTTP Request Headers (section [7.7.6](#)), and the `context` keyword in the query string, to disambiguate between multiple links of the same `linkType` (section [7.7.5](#))
19. SHOULD provide a resolver description file at `/.well-known/gs1resolver` (section [7.11](#))
20. If supporting multiple links per identified item, SHALL recognise the GS1 Web vocabulary namespace, noting its change management practice. A resolver SHOULD recognise the `schema.org` namespace. A resolver MAY recognise additional namespaces but link types defined elsewhere SHALL NOT duplicate link types available from GS1 and `schema.org`.
21. If supporting redirection based on a Global Company Prefix with no further processing taking place on the resolver, redirections SHALL be annotated with the `gs1:handledBy` link type (section [7.7.1](#)).
22. MAY provide content directly with no redirection (section [7.8](#)).
23. SHOULD tolerate trailing slashes at the end of GS1 Digital Link URIs, i.e. the resolver SHOULD NOT fail if one is present (section [9.4](#)).

7.13 GS1 Class Two resolver

The working group recognises that there are multiple possible methods by which identified objects can be linked to sources of information. This document specifies a particular method that combines the GS1 system with Web principles, including Linked Data. Existing and future systems may of course, take a different approach designed to meet other needs. In an effort to bridge the gap between such alternative systems and the GS1 Digital Link ecosystem, we define a GS1 Class Two resolver, which has the following characteristic:

When dereferencing a URI that identifies an item that can also be identified using the GS1 system, the response from the server SHALL include a conformant GS1 Digital Link URI in the HTTP Link header. The GS1 Digital Link URI SHOULD point to a fully conformant GS1 resolver.

8 Compression and decompression

This section and all subsections are normative except where indicated

8.1 Purpose of compression

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

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

8.2 Technical requirements extracted from the business requirements

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.

If the output of the compression algorithm still exceeds the capacity of the data carrier then an even shorter, unstructured URL may point to a class 2 resolver (section [7.13](#)) from which a conformant GS1 Digital Link may be obtained.

8.3 Viability of types of compression techniques and opportunities for compression

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

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

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

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

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

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

Using a full byte (8 bits) to represent a numeric digit uses more bits than the minimum needed; 4 bits are capable of encoding numeric digits 0-9 and hexadecimal characters a-f; approximately 3.32 bits per digit are required to encode a numeric string as a binary integer. This therefore represents a significant opportunity for compression particularly for all-numeric strings and dates.

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

8.4 Structure of the compressed string

8.4.1 Conversion between binary and URI-safe base 64 alphabet

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

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

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

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

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

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

AI	First Component			Second Component		
	Encoding E N = numeric, X=alphanumeric	Fixed Length L	Max. Length M	Encoding E N = numeric, X=alphanumeric	Fixed Length L	Max. Length M
00	N	18				
01	N	14				
02	N	14				
10	X		20			
11	N	6				
12	N	6				
13	N	6				
15	N	6				
16	N	6				
17	N	6				
20	N	2				
21	X		20			
22	X		20			
240	X		30			
241	X		30			
242	N		6			
243	X		20			
250	X		30			
251	X		30			
253	N	13		X		17
254	X		20			
255	N	13		N		12
30	N		8			
3100-3105 3110-3115 3120-3125 3130-3135 3140-3145	N	6				



3150-3155 3160-3165						
3200-3205 3210-3215 3220-3225 3230-3235 3240-3245 3250-3255 3260-3265 3270-3275 3280-3285 3290-3295	N	6				
3300-3305 3310-3315 3320-3325 3330-3335 3340-3345 3350-3355 3360-3365 3370-3375	N	6				
3400-3405 3410-3415 3420-3425 3430-3435 3440-3445 3450-3455 3460-3465 3470-3475 3480-3485 3490-3495	N	6				
3500-3505 3510-3515 3520-3525 3530-3535 3540-3545 3550-3555 3560-3565 3570-3575	N	6				
3600-3605 3610-3615 3620-3625 3630-3635 3640-3645 3650-3655 3660-3665 3670-3675 3680-3685 3690-3695	N	6				
37	N		8			
3900-3909	N		15			
3910-3919	N	3		N		15
3920-3929	N		15			
3930-3939	N	3		N		15
3940-3943	N	4				
400	X		30			



401	X		30			
402	N	17				
403	X		30			
410 - 416	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			
7001	N	13				
7002	X		30			
7003	N	10				
7004	N		4			
7005	X		12			
7006	N	6				
7007	N	6		N		6
7008	X		3			
7009	X		10			
7010	X		2			
7020	X		20			
7021	X		20			
7022	X		20			
7023	X		30			
7030-7039	N	3		X		27
710 - 714	X		20			
7230-7239	X		30			
8001	N	14				
8002	X		20			
8003	N	14		X		16

8004	X		30			
8005	N	6				
8006	N	18				
8007	X		24			
8008	N	8		N		4
8009	X		50			
8010	X		30			
8011	N		12			
8012	X		20			
8013	X		30			
8017	N	18				
8018	N	18				
8019	N		10			
8020	X		25			
8026	N	18				
8110	X		70			
8111	N	4				
8112	X		70			
8200	X		70			
90	X		30			
91-99	X		90			

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

The binary string consists of a concatenation of binary string encodings for each encoded key=value pair. For GS1 Application Identifiers, the key is usually the numeric GS1 Application Identifier. However, Table Opt (section 8.5) of the compression algorithm also defines 2-character hexadecimal Optimisation Codes that correspond to a pre-defined sequence of one or more GS1 Application Identifiers.

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

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

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

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

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

8.4.4 Length indicators and Encoding indicators

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

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

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

length alphanumeric component		
-------------------------------	--	--

8.4.4.1 Encoding indicator

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

Encoding value (decimal)	Encoding value (binary)	Character encoding
0	000	All-numeric string encoded as integer at ≈ 3.32 bits per character
1	001	Lower-case hexadecimal characters at 4 bits per character
2	010	Upper-case hexadecimal characters at 4 bits per character
3	011	URI-safe base64 characters A-Z a-z 0-9 hyphen and underscore at 6 bits per character
4	100	ASCII characters in range 0-127 at 7 bits per character
5	101	<i>reserved for other encoding (to be defined in future)</i>
6	110	<i>reserved for other encoding (to be defined in future)</i>
7	111	First Extension point to a longer encoding indicator (e.g. 6 bits) providing a further 7 values plus Second Extension point.

8.4.4.2 Length indicator

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

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

The length indicator consists of a number of bits, n , whose binary value corresponds to L where L is the actual length of the value of a variable-length value or value component. The number of bits (n) is selected depending of the maximum permitted length (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.

0	0	1	1	0
---	---	---	---	---

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

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

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

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

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

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

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

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

8.5 Optimised encoding of combinations of GS1 Application Identifiers

Capacity exists to support further compression of combinations of GS1 Application Identifiers. The table below lists a number of pre-defined sequence of GS1 Application Identifiers. Instead of encoding each numeric GS1 Application Identifier key using 4 bits per digit, a single 8 bit code indicates the pre-defined sequence.

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	<i>reserved</i>	<i>6 values reserved for future use</i>	
50 – 5F	<i>reserved</i>	<i>16 values reserved for future use</i>	
60 – 6F	<i>reserved</i>	<i>16 values reserved for future use</i>	
70 – 7F	<i>reserved</i>	<i>16 values reserved for future use</i>	
80 – 8F	<i>reserved</i>	<i>16 values reserved for future use</i>	
9E , 9F	<i>reserved</i>	<i>2 values reserved for future use</i>	
AA – AF	<i>reserved</i>	<i>6 values reserved for future use</i>	
B0 – BF	<i>reserved</i>	<i>16 values reserved for future use</i>	
CE , CF	<i>reserved</i>	<i>2 values reserved for future use</i>	
D0 – DF	<i>reserved</i>	<i>Reserved for expressing version number of compression algorithm</i>	
E0 – EF	<i>reserved</i>	<i>16 values reserved for non-collision with non-standard compression algorithms</i>	Means that no GS1 standard compressed string SHALL begin with a 4-7
F0 – FF	<i>reserved</i>	<i>reserved for indicating non-GS1 key:value pairs within compressed string</i>	
		<i>= 112 values reserved for future use</i>	

8.6 Examples of binary encoding of GS1 Application Identifiers

This section is informative

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

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

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

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

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

8.6.1.1 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	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	0	1	1	1	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	1	1

8.6.1.2 Binary encoding example for Expiration Date and Time: (7003)190311228

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		190311228																																				
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	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

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

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

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

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

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

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

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

8.6.2.1 Binary encoding example for Count: (30)5000

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

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

The following examples show binary encoding of GS1 Application Identifiers where the value is permitted to be alphanumeric and of variable length. In this situation, the GS1 Application Identifier (encoded at four bits per digit) is immediately followed by a 3-bit Encoding Indicator whose value indicates whether the actual value can be considered as all-numeric, lower-case hexadecimal, upper-case hexadecimal, characters from a URI-safe base 64 alphabet or ASCII characters. The value of the 3-bit Encoding Indicator is populated according to the table in section 8.4.4.1. The 3-bit Encoding Indicator is then followed by a Length Indicator consisting of N bits that indicate the length of the actual value in digits or characters.

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

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

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

8.6.3.2 Batch/Lot Number: (10)XYZ98765

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

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

8.6.3.3 Batch/Lot Number: (10)ABC98765

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

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

8.6.3.4 Batch/Lot Number: (10)12398765

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

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

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

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

8.6.4 Optimisation using pre-defined sequences of GS1 Application Identifiers

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

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



Code: 0000 1110
 Value: 00010011010011101001100000111000111101110000011
 Value: 00011100010110111100101100001111000

Optimisation Code 'OE' = (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 1 0 0 1 1 1 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1	0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1	0 0 0 1 1 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 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 'OE' 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>).

8.7 Formal ABNF grammar for compressed GS1 Digital Link URIs

Section [6](#) defines the formal ABNF grammar for uncompressed GS1 Digital Link URIs. This section extends section [6](#) 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 [8.4.1](#).

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

```
compressedSegment = 1*uriSafeBase64char
```

Section [8.7.1](#) defines the formal ABNF grammar for partially compressed GS1 Digital Link URIs.

Section [8.7.2](#) defines the formal ABNF grammar for fully compressed GS1 Digital Link URIs.

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

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

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

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

After removal of any trailing forward slash from the URI path information, it is possible to count backwards (from right to left) through the URI path components that are separated via forward slash characters. Counting the final component of the URI path information as component 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 [8.9](#) and flowcharts D1-D4.

8.7.1 Partially compressed GS1 Digital Link URIs

A partially compressed GS1 Digital Link URI includes exactly one primary GS1 identification key and its value in uncompressed format within the URI path information. The final component of the URI path information is a compressed segment consisting of one or 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 [5.6](#).

```
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 / ssc-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
```

8.7.2 Fully compressed GS1 Digital Link URIs

A fully compressed GS1 Digital Link URI includes a compressed segment as the final component of its URI path information. The final component of the URI path information is a compressed segment consisting of one or 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]

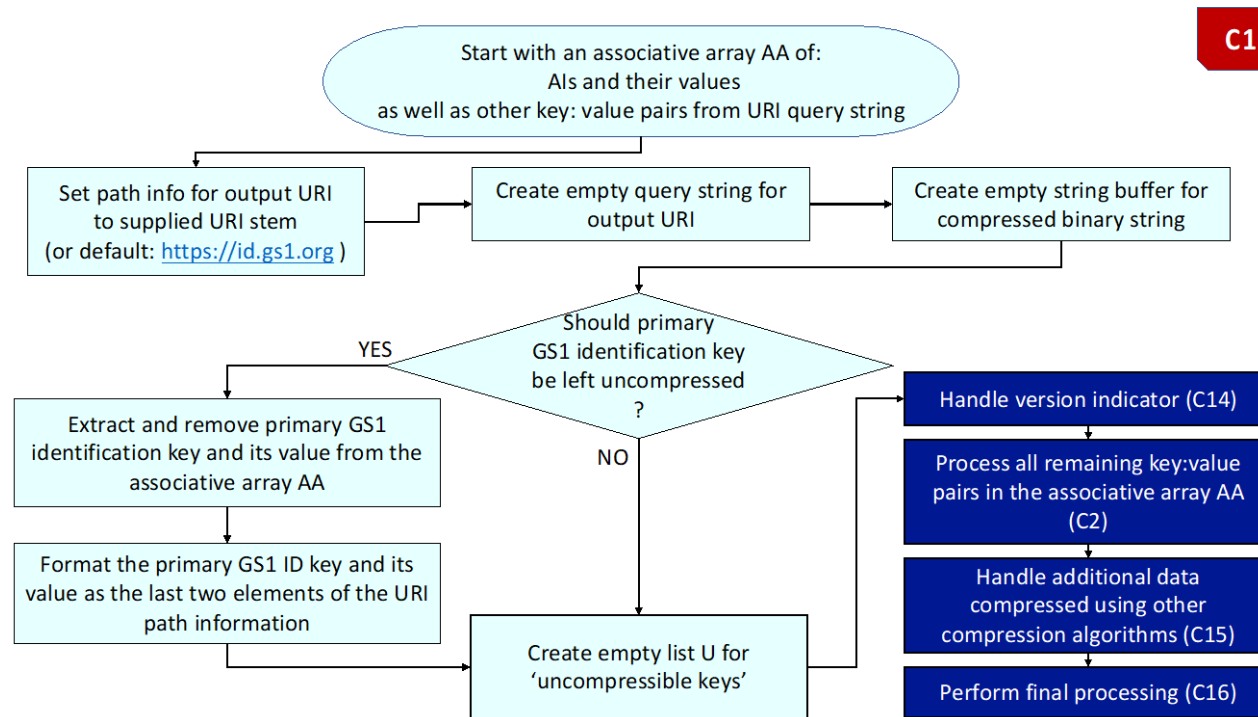
fullyCompressedReferenceGS1webURI
    = "https://id.gs1.org" fullyCompressedGS1webURIPattern

fullyCompressedCustomGS1webURI
    = customURIStem fullyCompressedGS1webURIPattern
```


8.8 Compression procedure and flowcharts

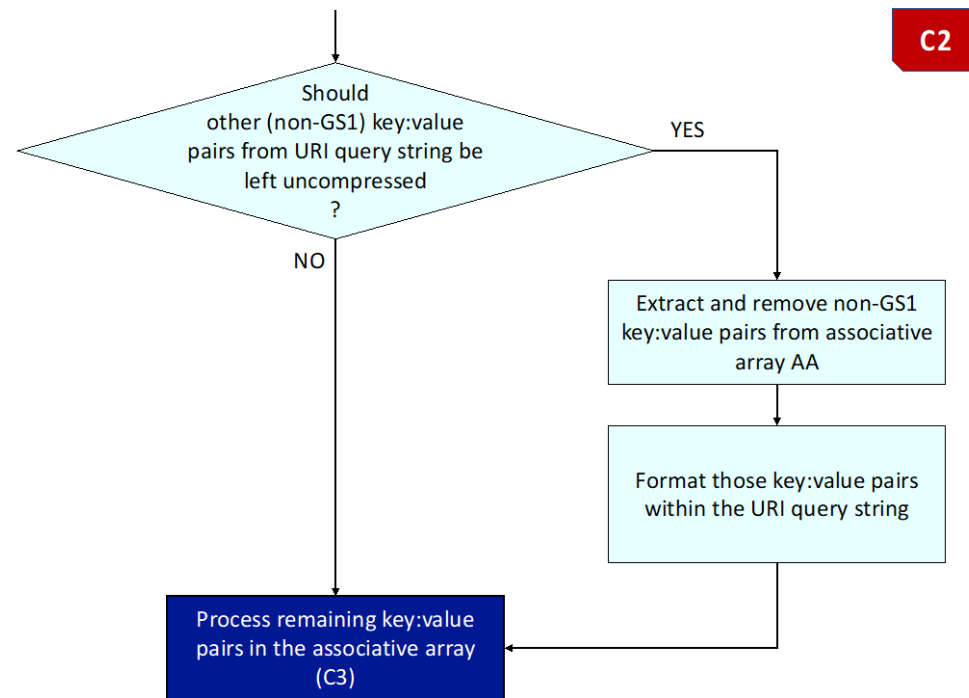
This section provides a set of flowcharts to describe the compression procedure to obtain fully or partially compressed GS1 Digital Link URIs.

Figure 8-1 C1: Top-level flowchart for v1.0 standard compression of GS1 Digital Link URIs



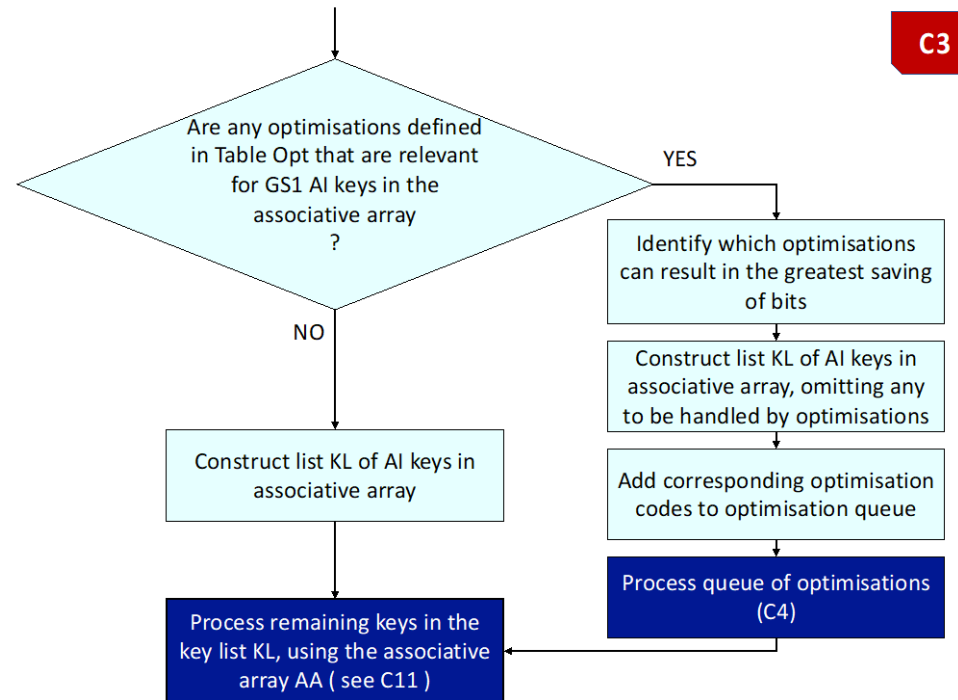
Flowchart C1 provides the high-level flowchart for compression of GS1 Digital Link URIs, starting with an associative array of key:value pairs from element strings or other URI query string key:value pairs. The first decision is whether the primary identifier and its value should be left uncompressed (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 8-2 C2 : Support non-GS1 key:value pairs from URI query string being compressed or left uncompressed in the URI query string



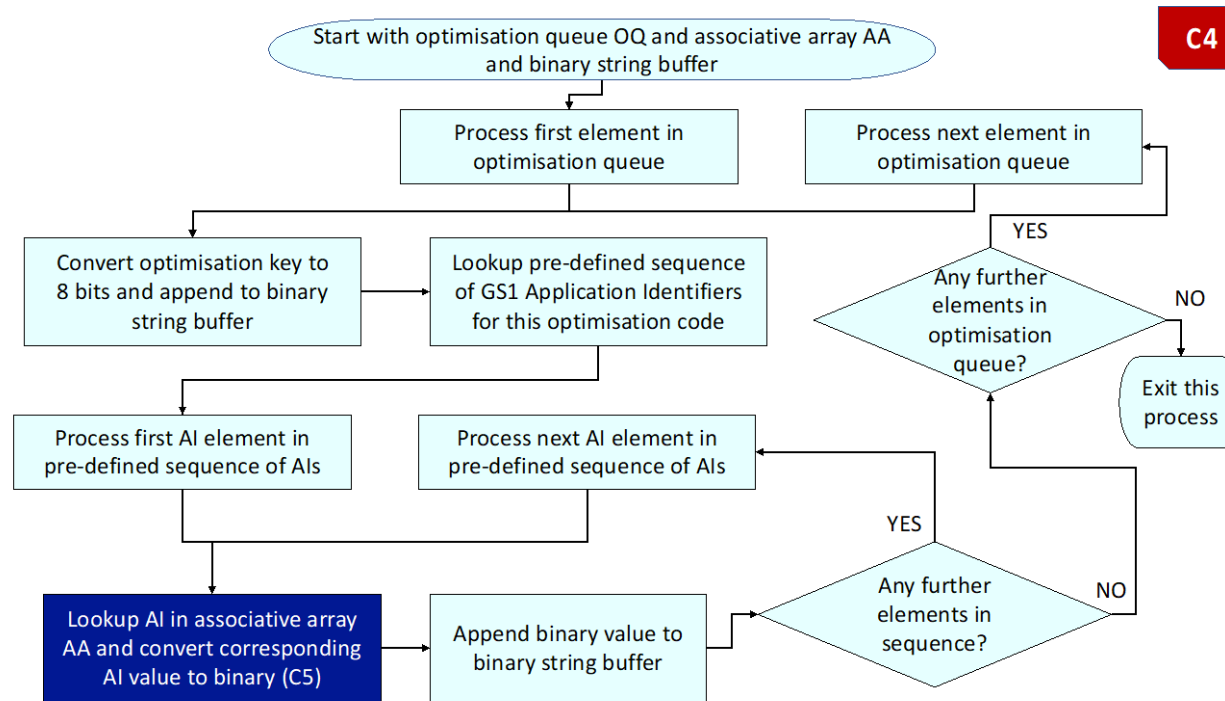
Flowchart C2 explains that if non-GS1 key:value pairs from the URI query string should be compressed, then these must be extracted from the URI query string and included within the associative array. Further processing then moves to flowchart C3.

Figure 8-3 C3: Support for optimisations using pre-defined sequences of GS1 Application Identifiers – refers to flowchart C4 for further detail

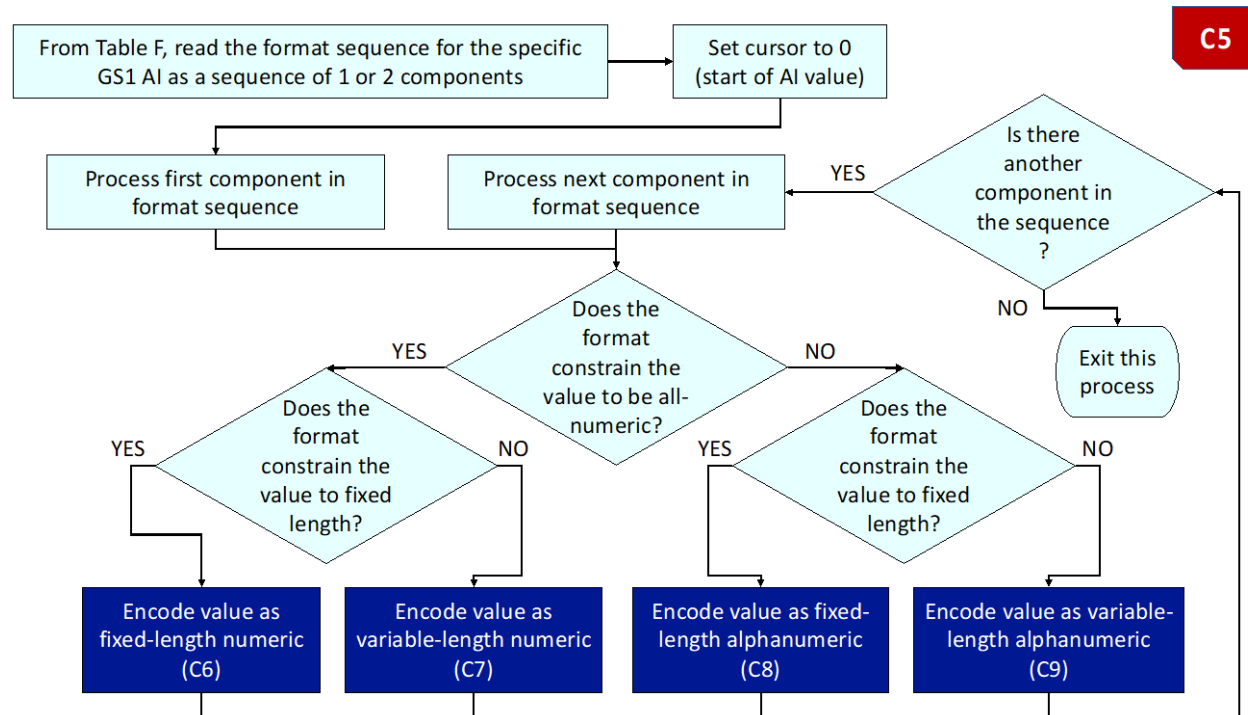


In Flowchart C3, the keys within the associative array are compared against available optimised pre-defined sequences of GS1 AIs, defined in Table Opt. If such optimisations are found, the combination of optimisations that results in the largest total saving of bits is selected and an updated list of 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.

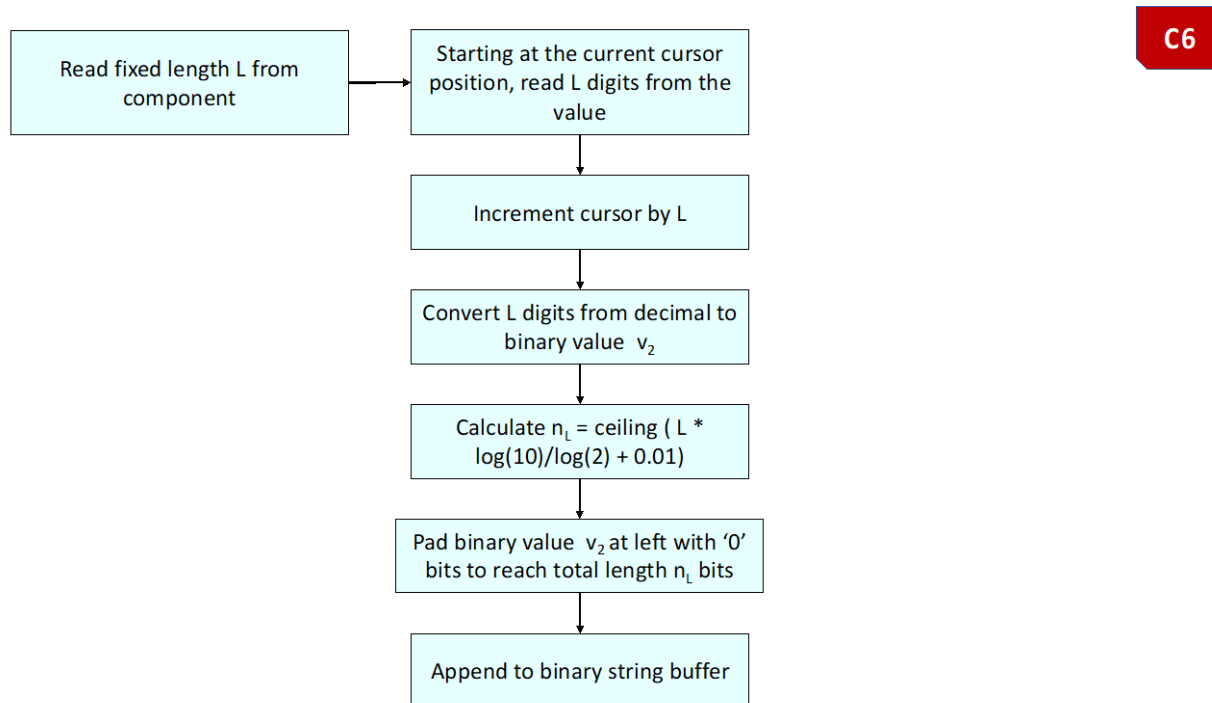
Figure 8-4 C4: Process queue of optimisations for pre-defined sequences of GS1 Application Identifiers



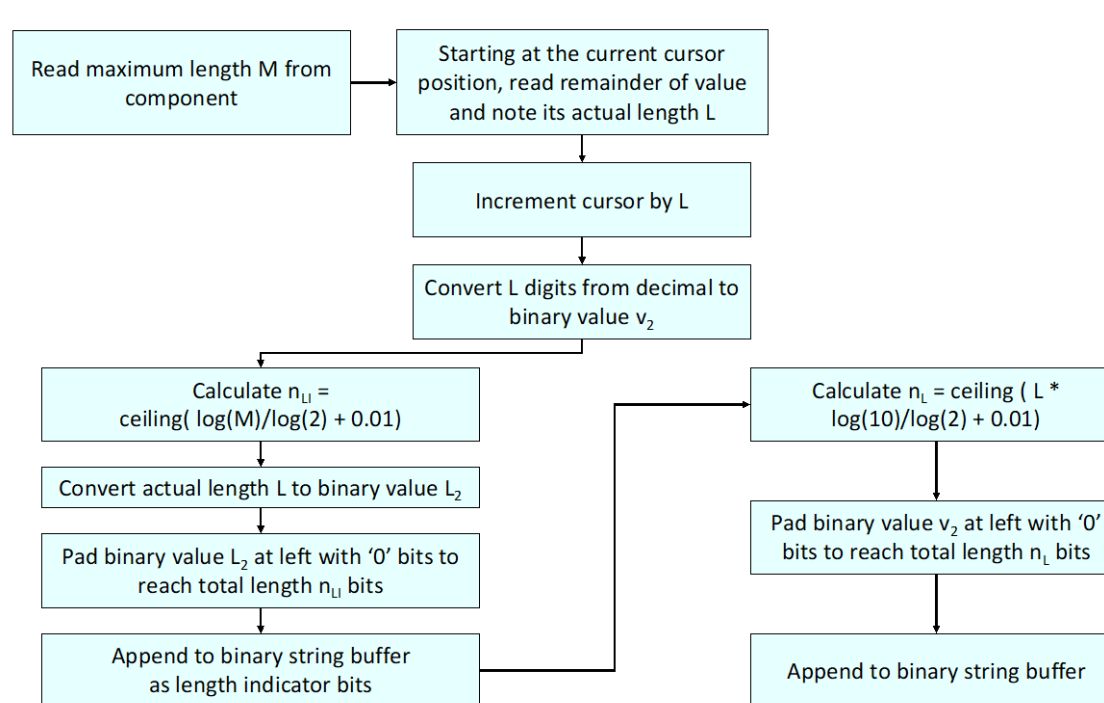
Flowchart C4 explains processing of a queue of optimisations for pre-defined sequences of GS1 Application Identifiers. For each optimisation, the GS1 Application Identifiers are not encoded in binary using 8, 12 or 16 bits per GS1 AI key. Instead a single 8-bit key is used to represent the entire pre-defined 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 8-5 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.

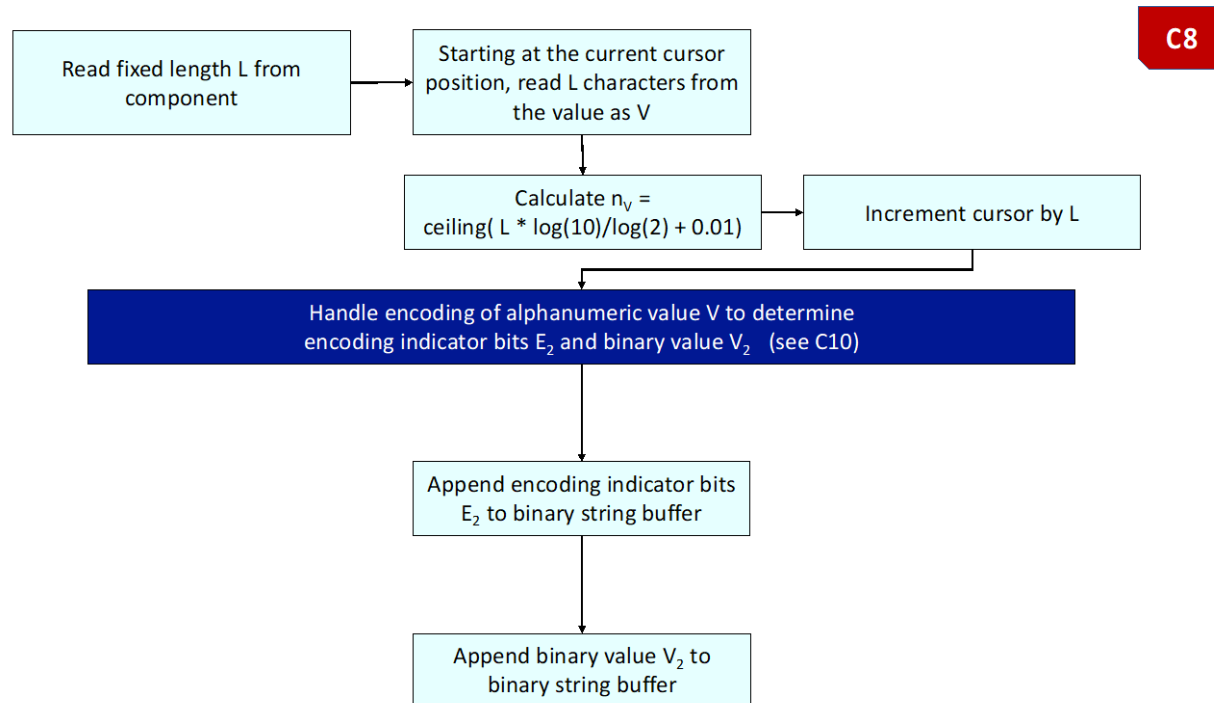
Figure 8-6 C6 : Encode a fixed-length numeric value


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 8-7 C7: Encode a variable-length numeric value


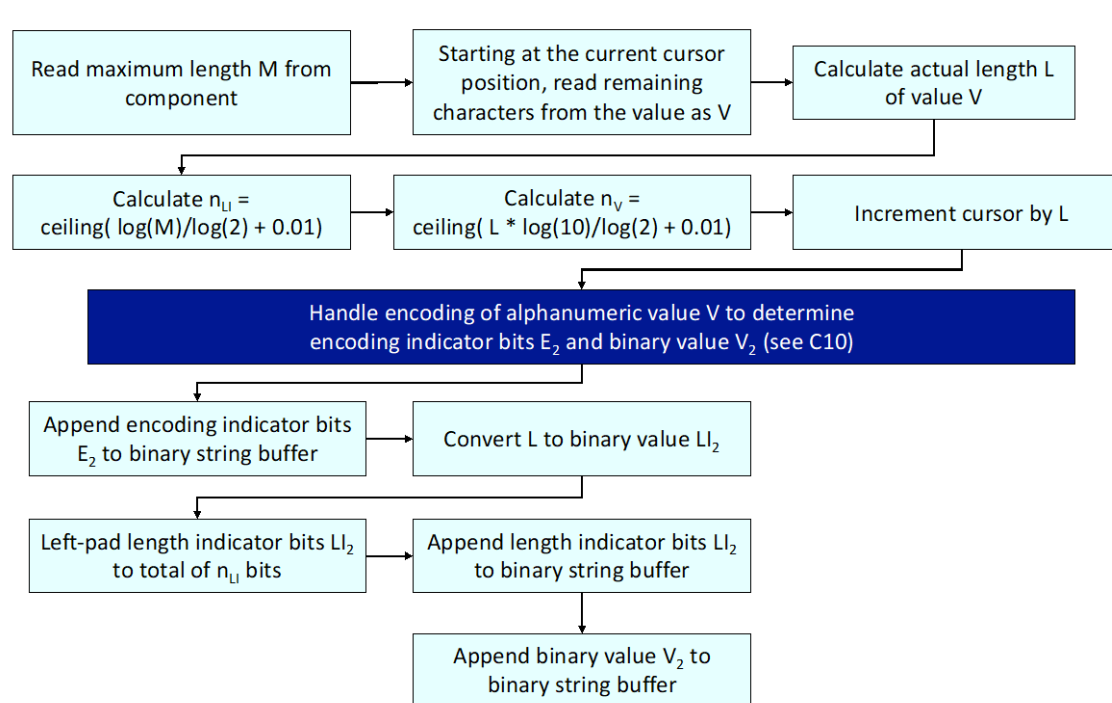
Flowchart C7 explains how to encode a variable-length numeric value. Firstly, a length indicator of an appropriate number of bit n_{LI} is encoded, according to the formula for n_{LI} , encoding the actual length L of the value. Next, the actual value is encoded as a binary integer that is left-padded 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 8-8 C8: Encode a fixed-length alphanumeric value, referring to flowchart C10 for additional details



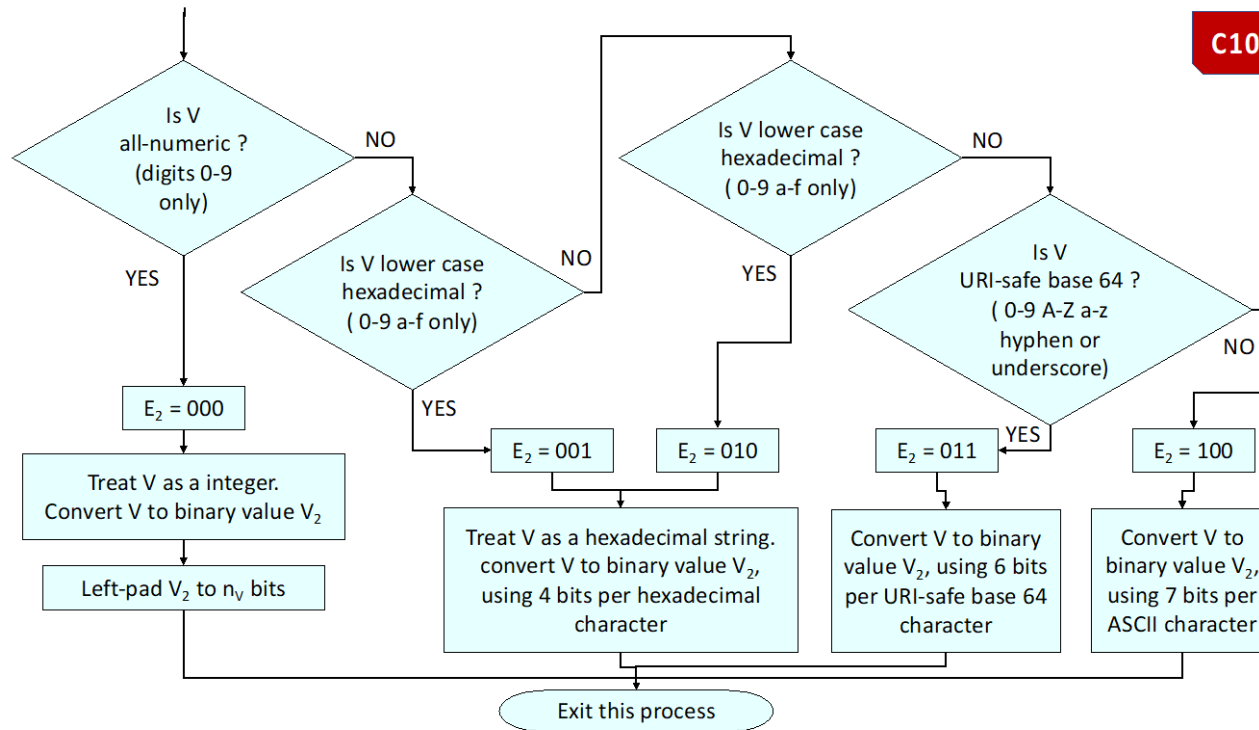
Flowchart C8 explains how to encode a fixed-length alphanumeric value. Flowchart C10 is used to determine an appropriate 3-bit encoding indicator E_2 which is encoded in the binary string before encoding the actual value binary value V_2 , using an appropriate number of bits depending on its length and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

Figure 8-9 C9: Encode a variable-length alphanumeric value, referring to flowchart C10 for additional details



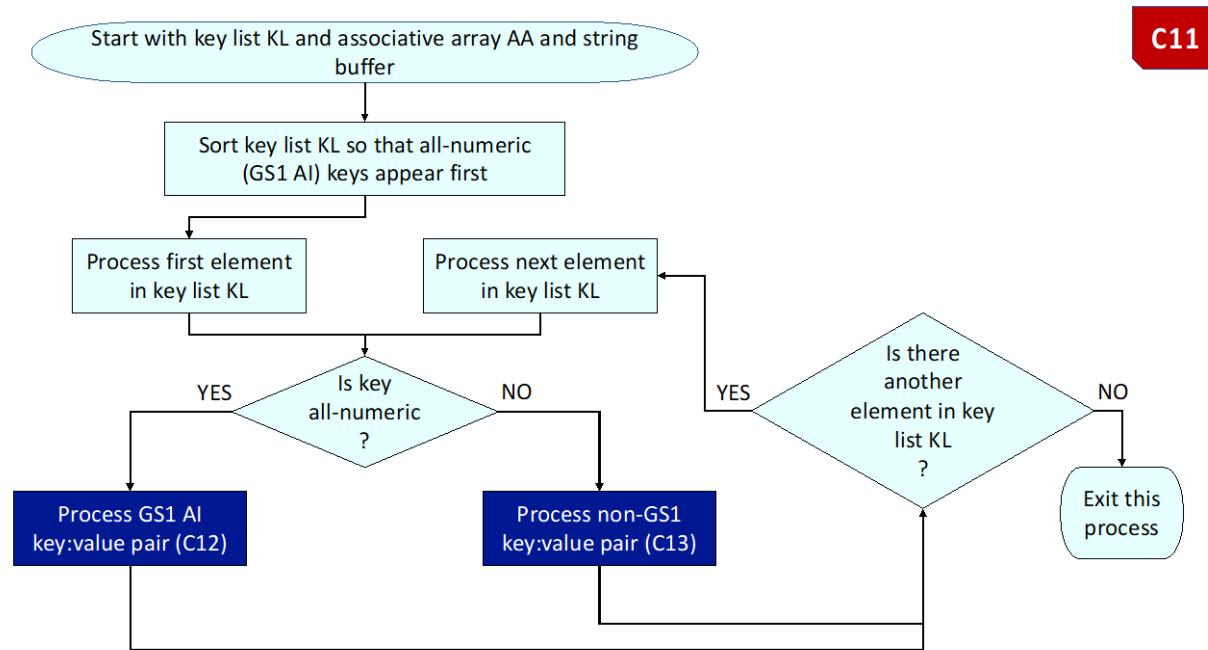
Flowchart C9 explains how to encode a variable-length alphanumeric value. This flowchart combines aspects of flowcharts C7 (variable-length, length indicator) and C8 (encoding indicator). As in Flowchart C9, Flowchart 10 is used to determine the 3-bit encoding indicator E_2 and the appropriate binary 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 8-10 C10 : Handle binary encoding of alphanumeric values



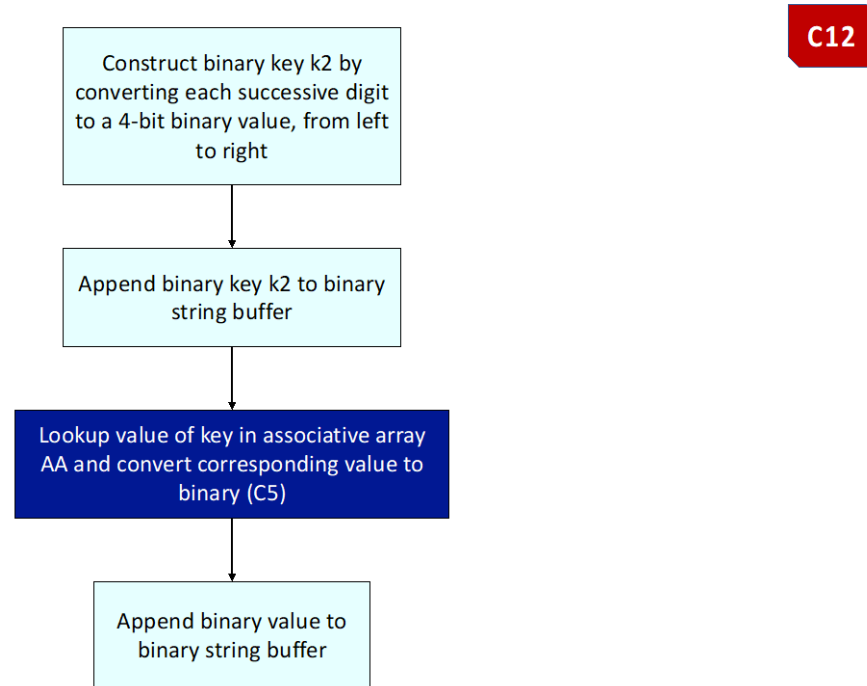
Flowchart C10 explains how to handle binary encoding of an alphanumeric value depending on whether the actual value is all-numeric, lower-case hexadecimal, upper-case hexadecimal, URI-safe base 64 or ASCII. For each of these, an appropriate 3-bit encoding indicator E2 is determined and the actual value V is converted to binary value V₂ 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 8-11 C11: Process remaining keys in the key list KL, using the associative array AA

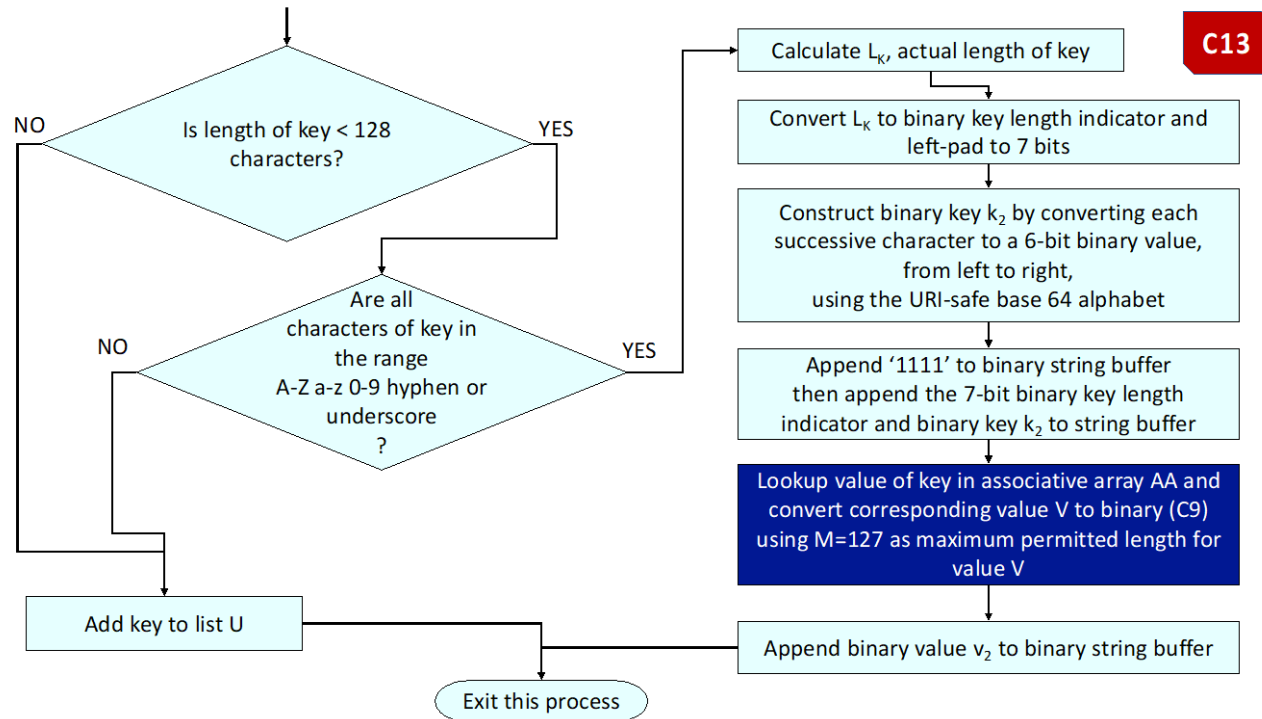


Flowchart C11 explains the processing of each key in the key list KL. The value of each key can be retrieved from the associative array AA. If the key is all-numeric, further processing references Flowchart C12 for processing of a GS1 AI key:value pair. If the key is not all-numeric, further processing 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.

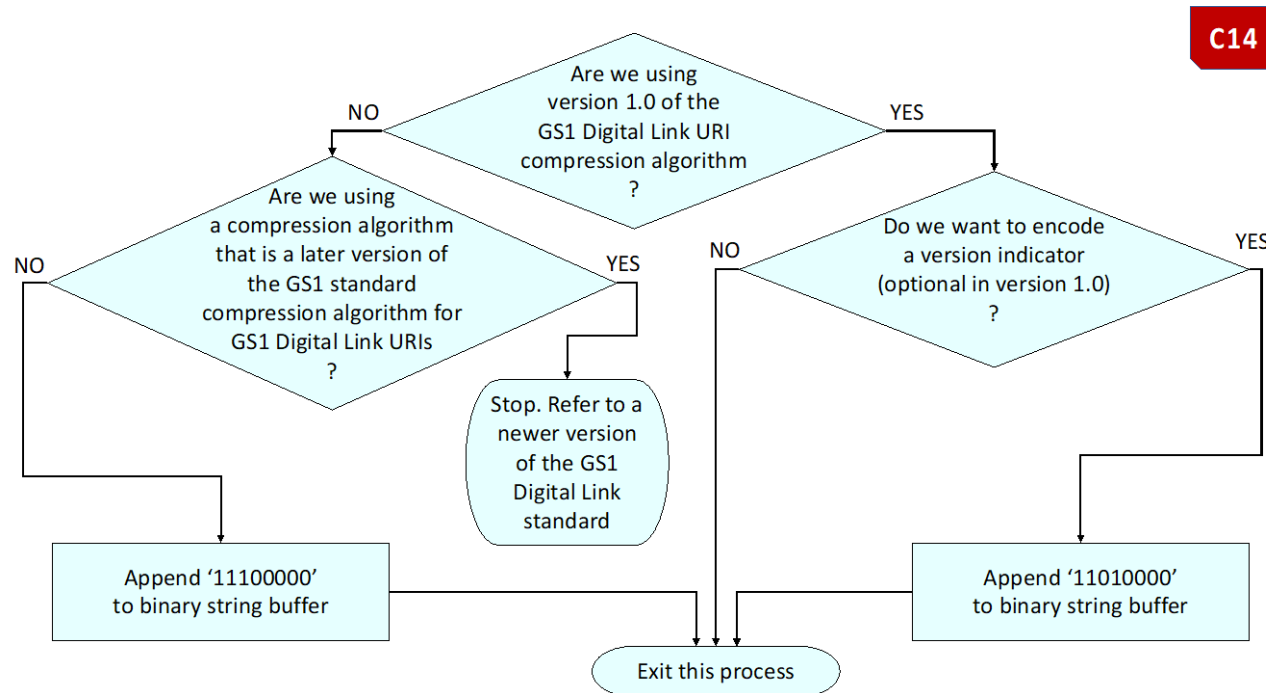
Figure 8-12 C12 : Encode GS1 Application Identifier key at 4 bits per digit, then encode the corresponding value in binary, referring to flowchart C5



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.

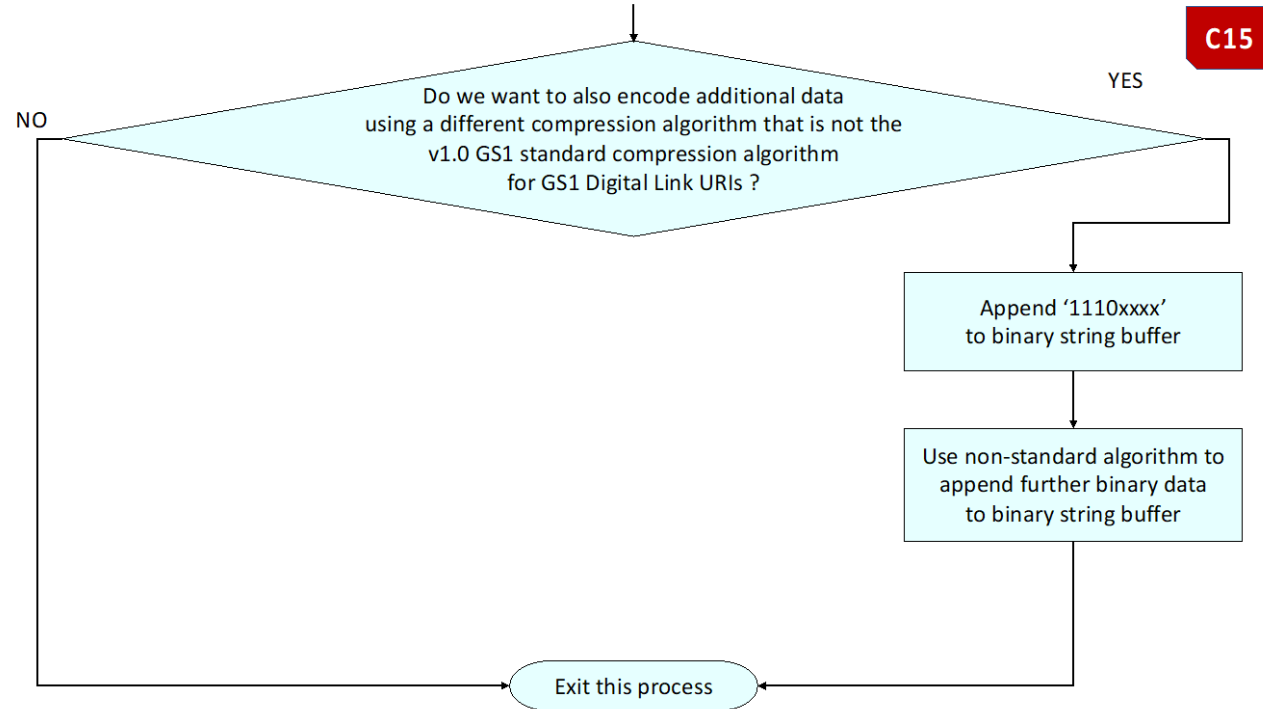
Figure 8-13 C13: Support compression of other non-GS1 key:value pairs from URI query string


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.

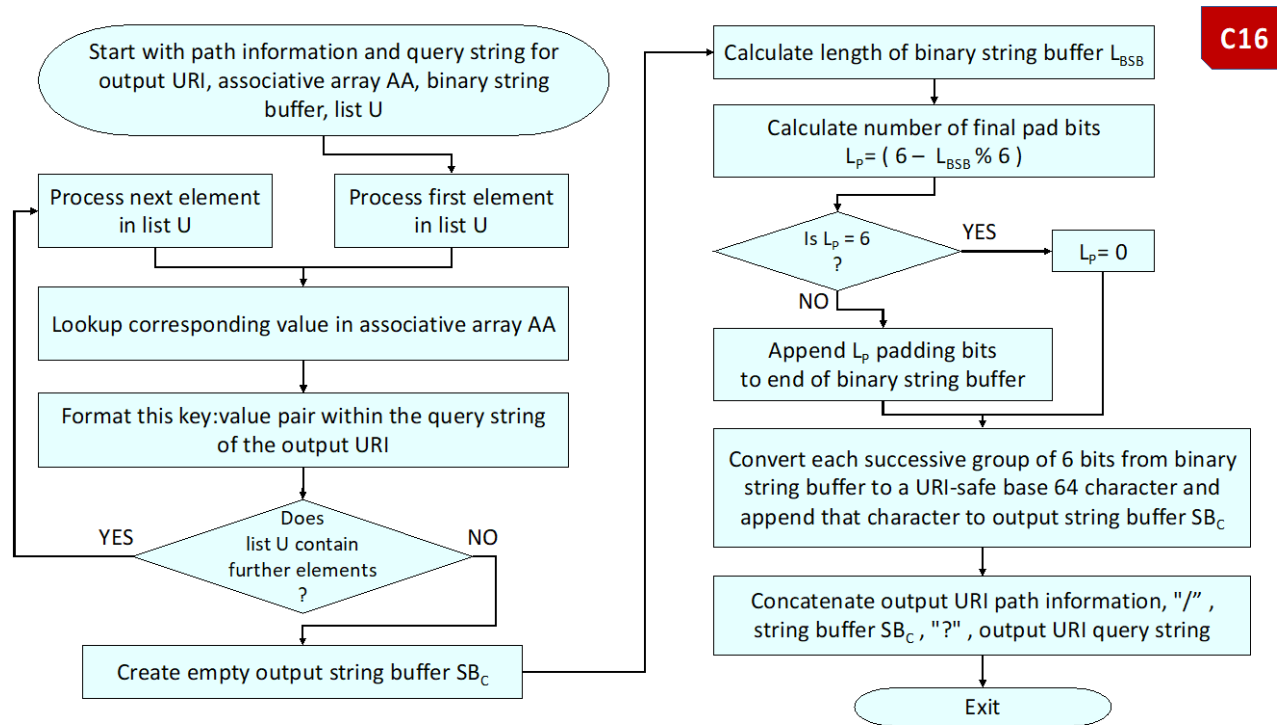
Figure 8-14 C14: Handle version indicator (optional in v1.0)


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 8-15 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.

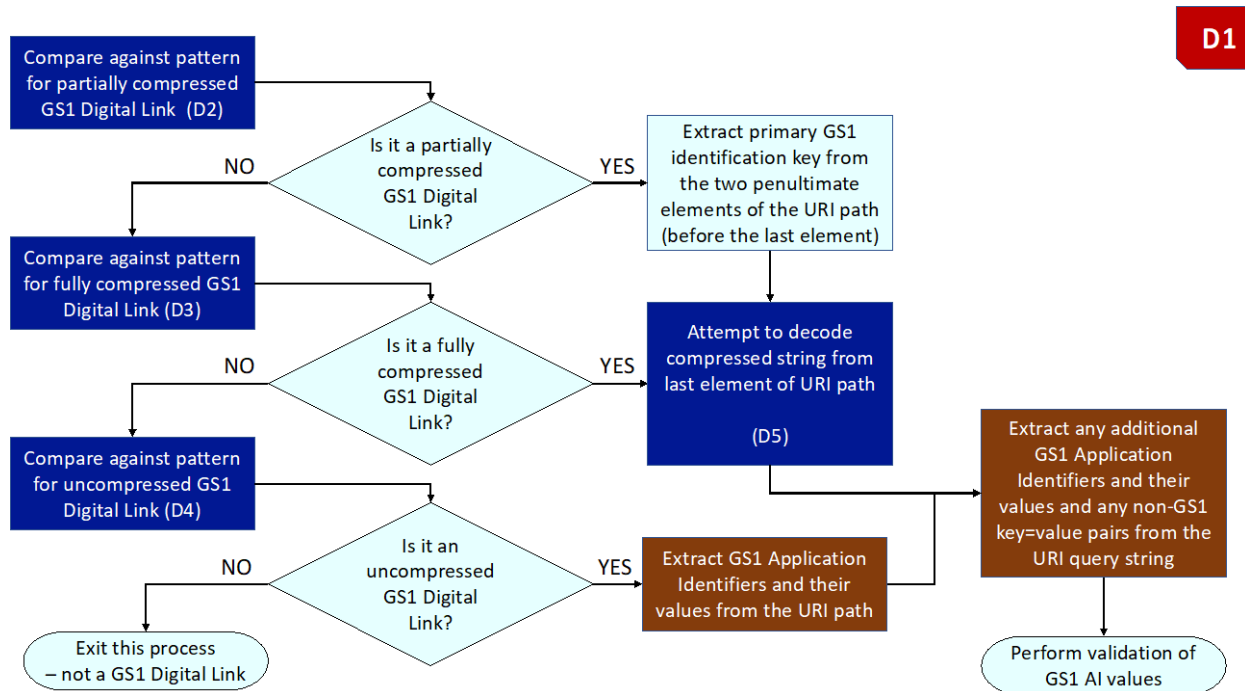
Figure 8-16 C16: Final assembly of compressed GS1 Digital Link URI


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.

8.9 Decompression procedure and flowcharts

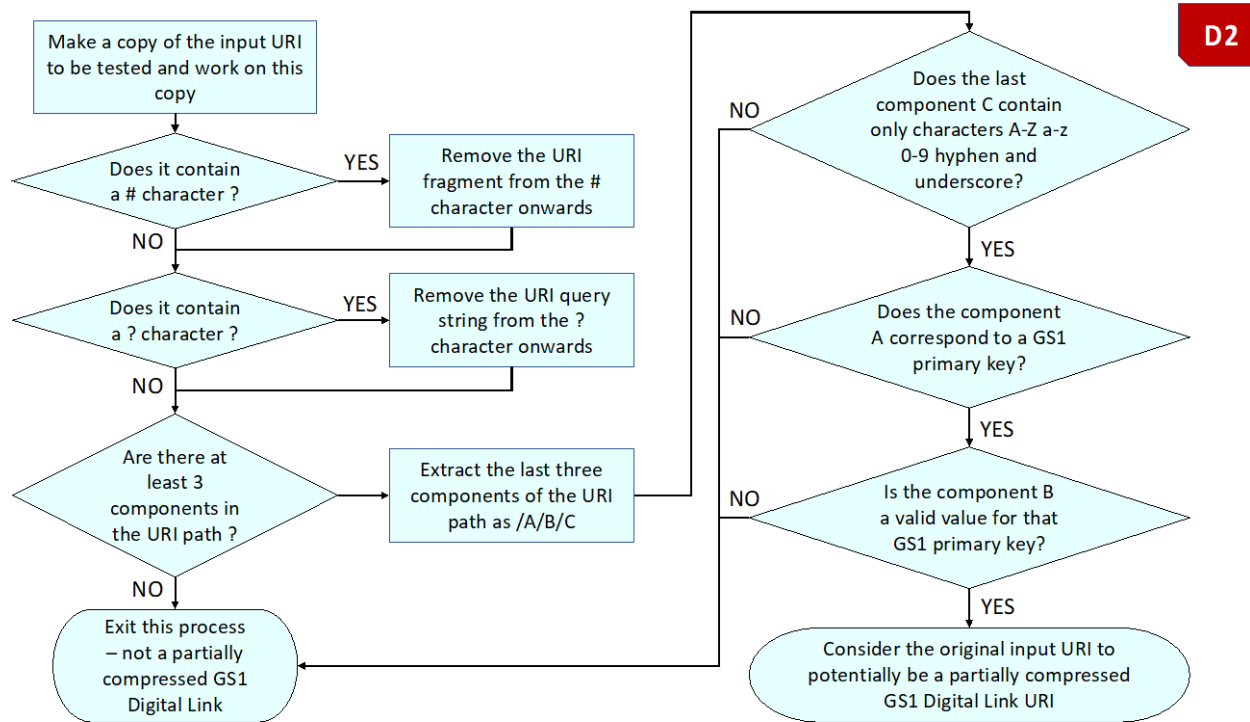
This section provides a set of flowcharts to describe the decompression procedure for fully or partially compressed GS1 Digital Link URIs. The result is an associative array of key:value pairs that can be translated to an uncompressed GS1 Digital Link URI.

Figure 8-17 D1: Top-level decompression flowchart



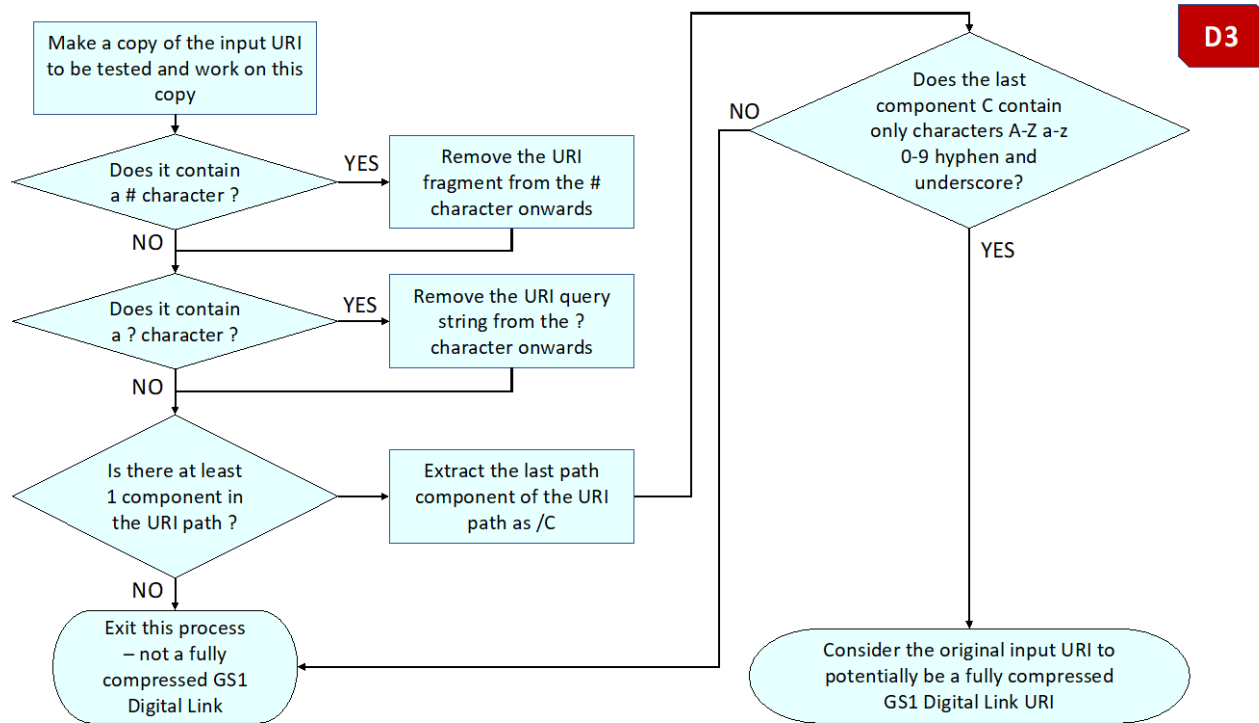
The decompression procedure begins by determining which kind of GS1 Digital Link URI is involved – uncompressed, partially compressed or fully compressed. Flowcharts D2, D3 and D4 are used to compare against patterns for partially compressed, fully compressed or uncompressed GS1 Digital Link URIs. For 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 8-18 D2: Compare against pattern for partially compressed GS1 Digital Link URI

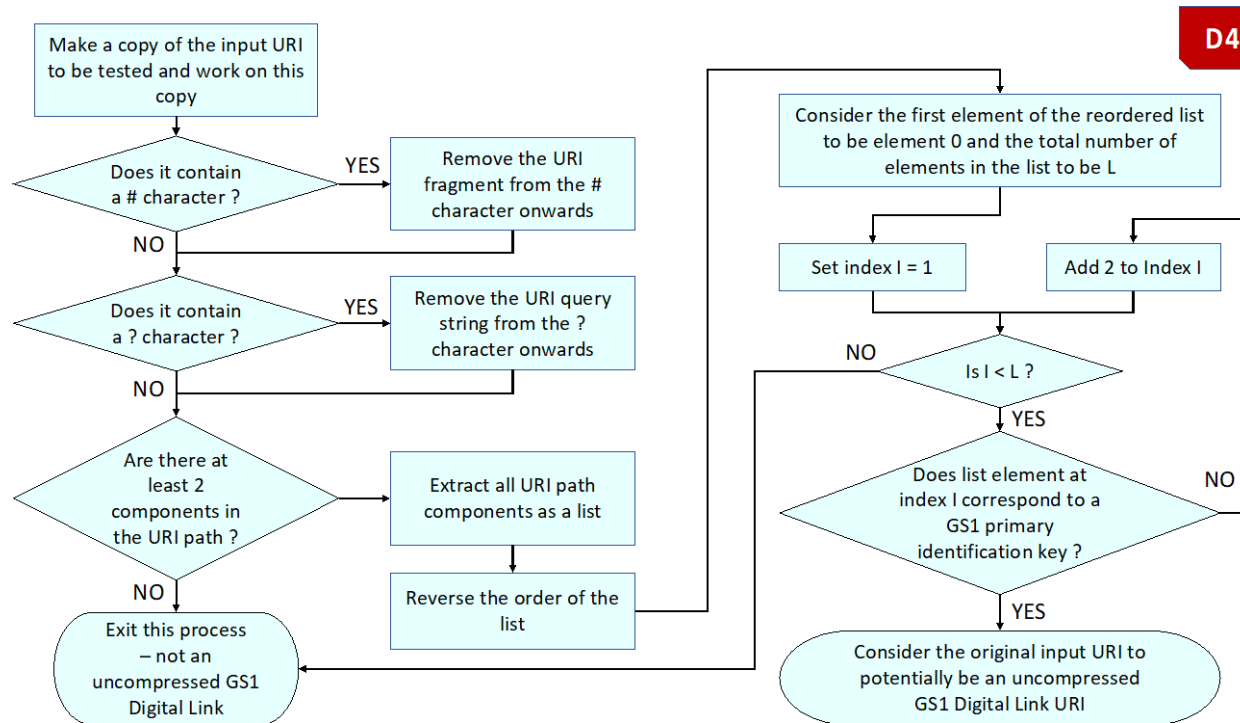


Flowchart D2 explains how to check for a partially compressed GS1 Digital Link URI in which the last component of the URI path consists of only characters from the URI-safe base 64 alphabet (A-Z a-z 0-9 hyphen and underscore) and is preceded by two URI path components, the first of which must correspond to a primary GS1 identification key such as GTIN, expressed either using numeric GS1 Application Identifiers or using the alphabetic short names defined in section 5.9.

Figure 8-19 D3: Compare against pattern for fully compressed GS1 Digital Link URI

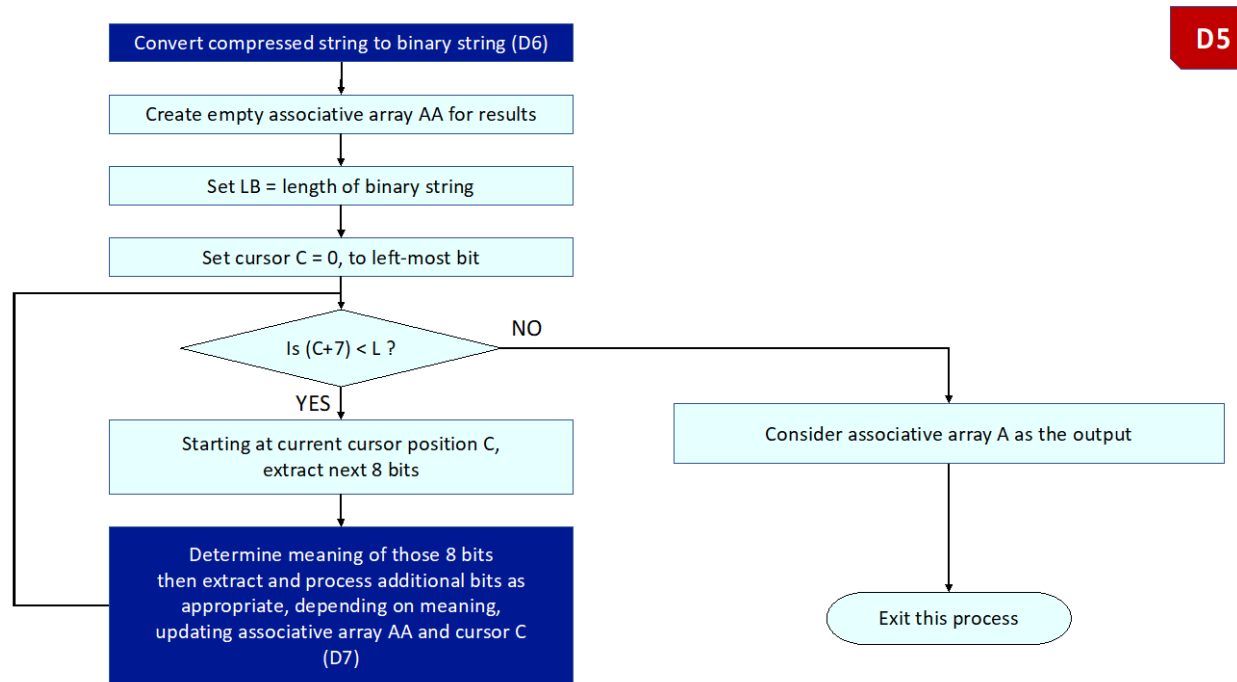


Flowchart D3 explains how to check for a partially compressed GS1 Digital Link URI in which the last component of the URI path consists of only characters from the URI-safe base 64 alphabet (A-Z a-z 0-9 hyphen and underscore). Note that because this pattern would also match a partially compressed GS1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

Figure 8-20 D4: Compare against pattern for uncompressed GS1 Digital Link URI


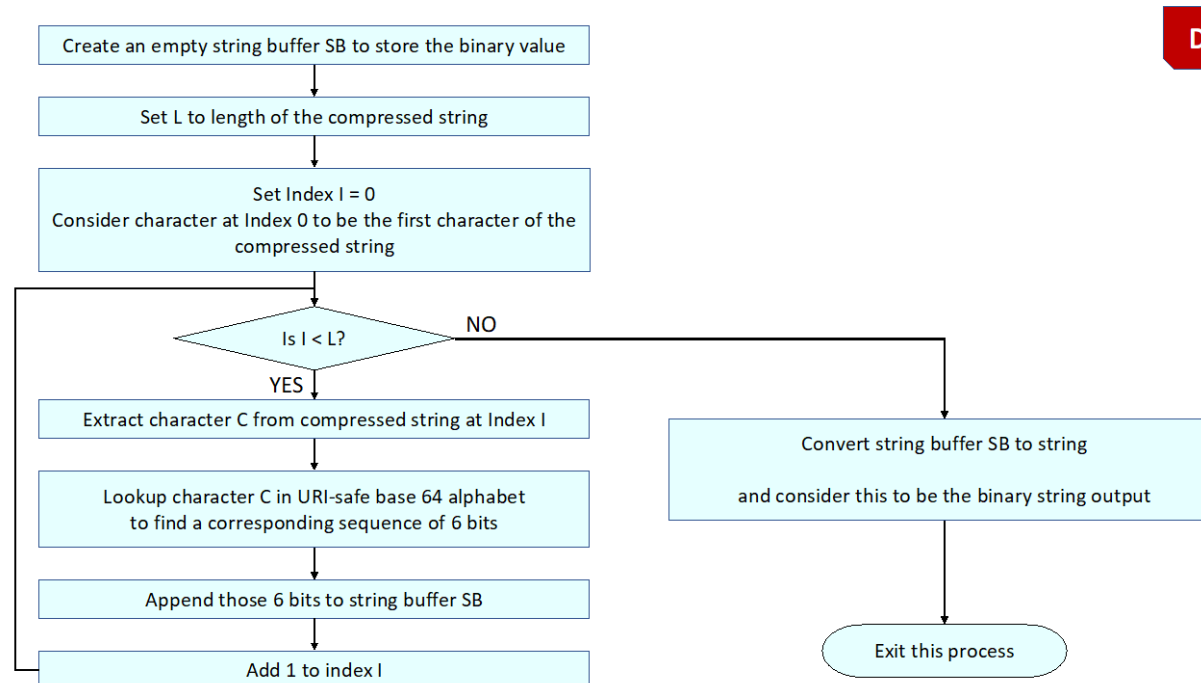
Flowchart D4 explains how to check for an uncompressed GS1 Digital Link URI. The URI path information can be analysed from right to left, considering two components at a time, testing whether the first of the pair of components corresponds to a primary GS1 identification key such as GTIN. Note that this 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 8-21 D5: High-level decoding flowchart, references flowcharts D6 (convert URI-safe base 64 characters to binary) and D7 (extract meaning and populate associative array)



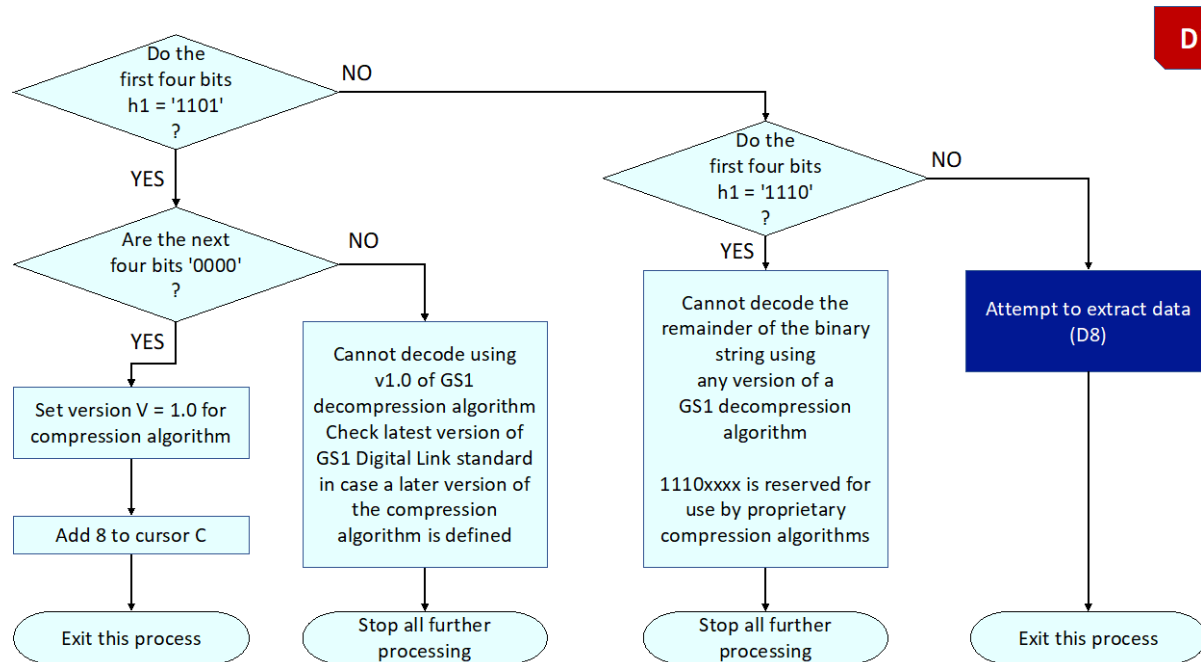
Flowchart D5 is the high-level flowchart for decoding the compressed string that is the final URI path component of a partially or fully compressed GS1 Digital Link URI. It references Flowchart D6 for the conversion from the URI-safe base 64 alphabet characters into a binary string. The main processing 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 8-22 D6: Convert the compressed string from URI-safe base 64 characters to a binary string

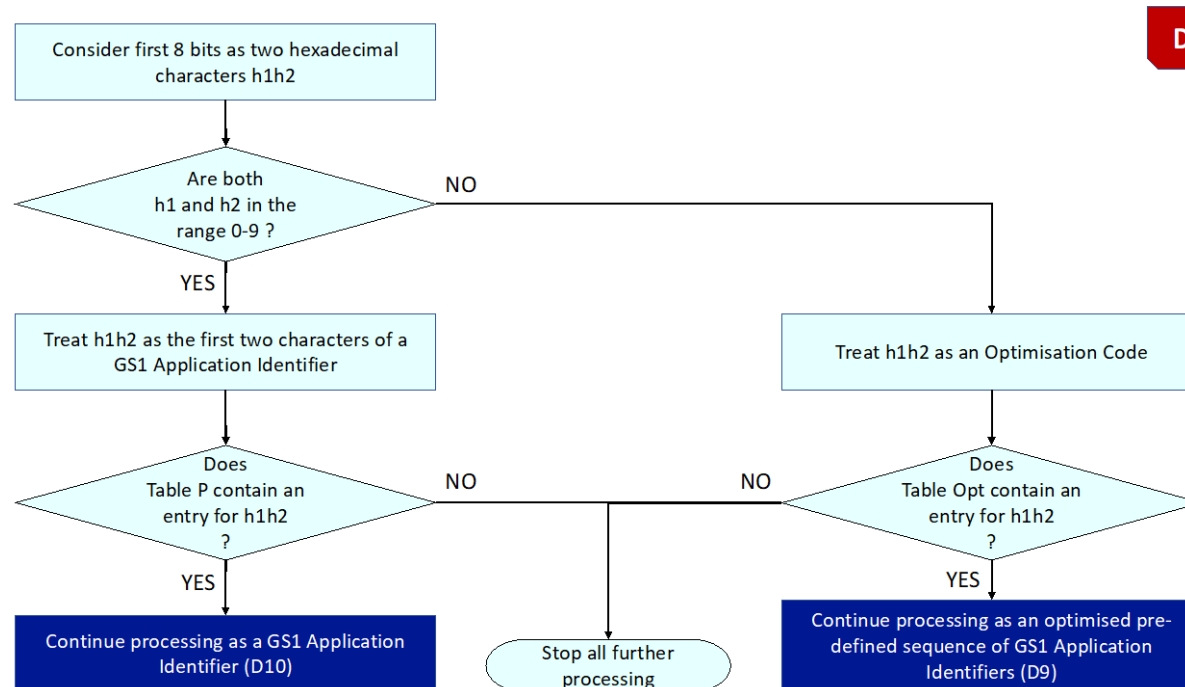


Flowchart D6 explains how to convert the compressed string expressed using URI-safe base 64 characters into a binary string. Each character encodes a sequence of 6 bits. The compressed string is decoded from left to right and the corresponding set of 6 bits per URI-safe base 64 character are also appended 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 8-23 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

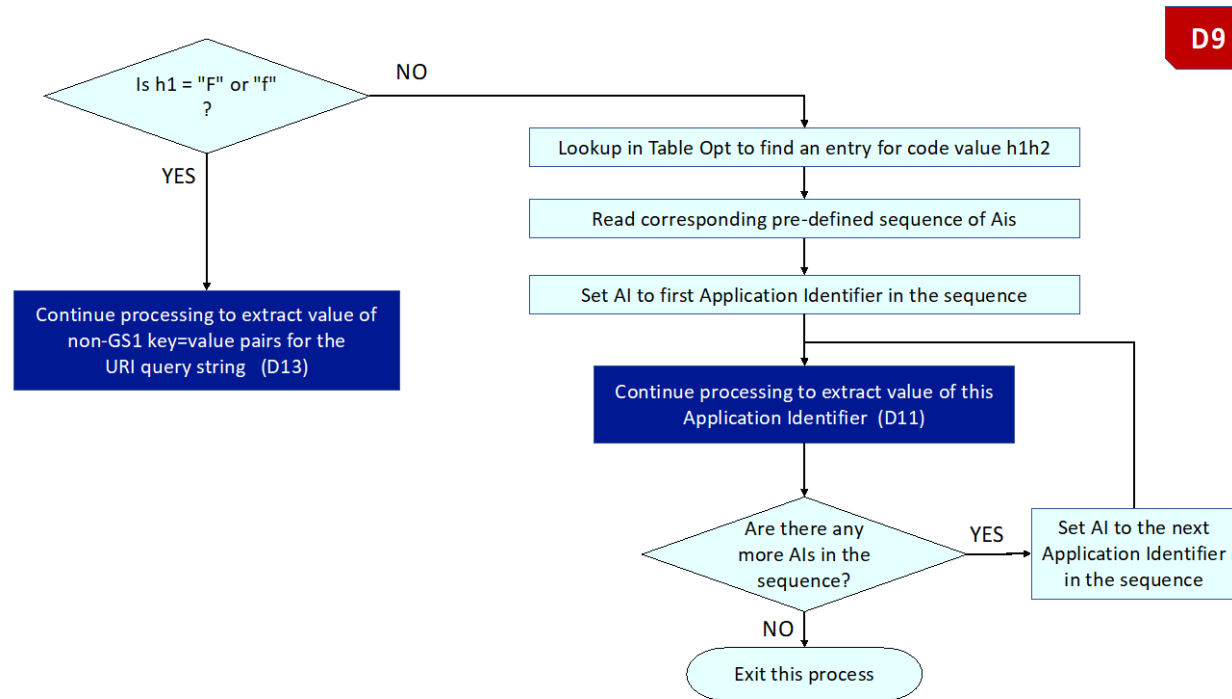


Flowchart D7 explains a sequence of checks on each 8-bit sequence read from the binary string on each iteration of the main loop of Flowchart D5. This includes detection of a potential version number. Currently only version 1.0 of the GS1 decomposition algorithm is defined and identified as binary sequence 11010000 (D0 in hexadecimal). If a pattern of 1110xxxx is encountered, this indicates a handover / extension point to a non-standard decomposition algorithm and the GS1 decomposition 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.

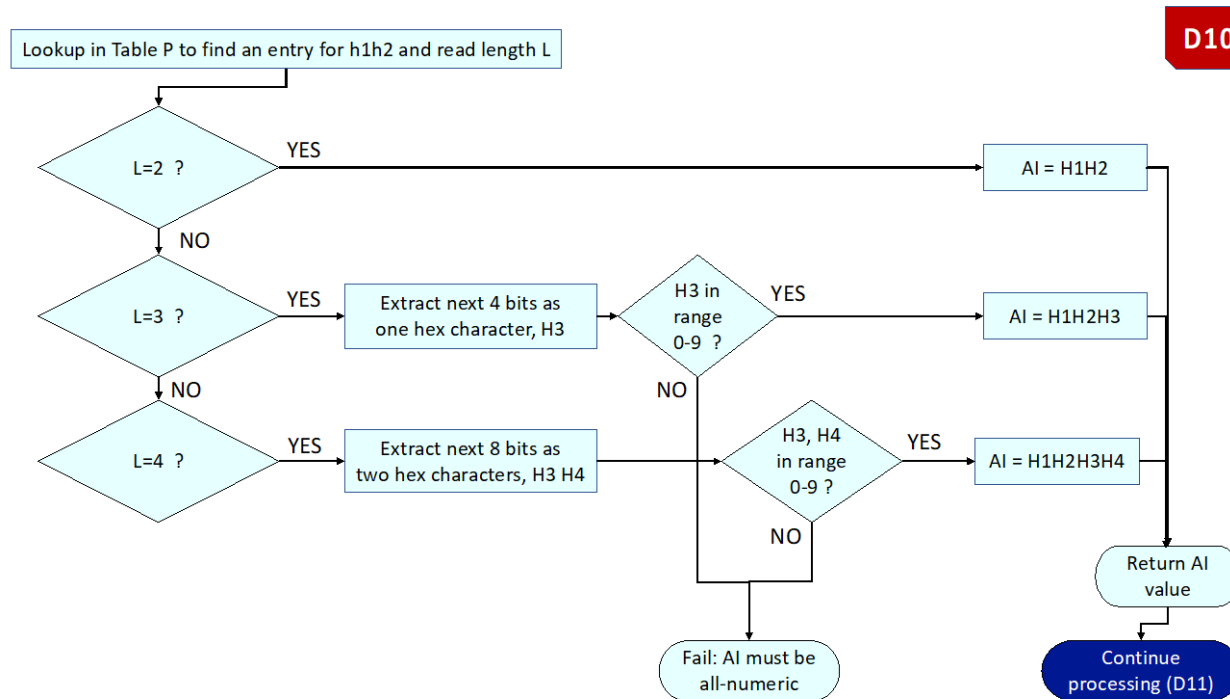
Figure 8-24 D8: Extract data from binary string


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.

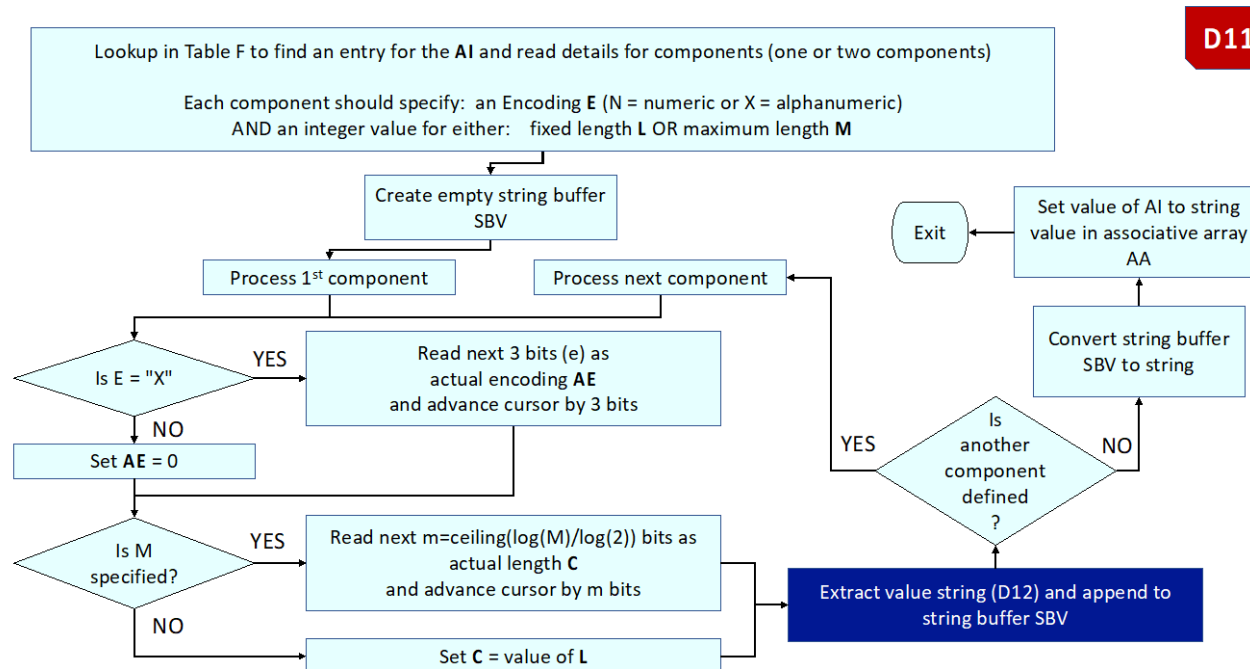
Figure 8-25 D9: Processing as an optimised pre-defined sequence of GS1 Application Identifiers



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.

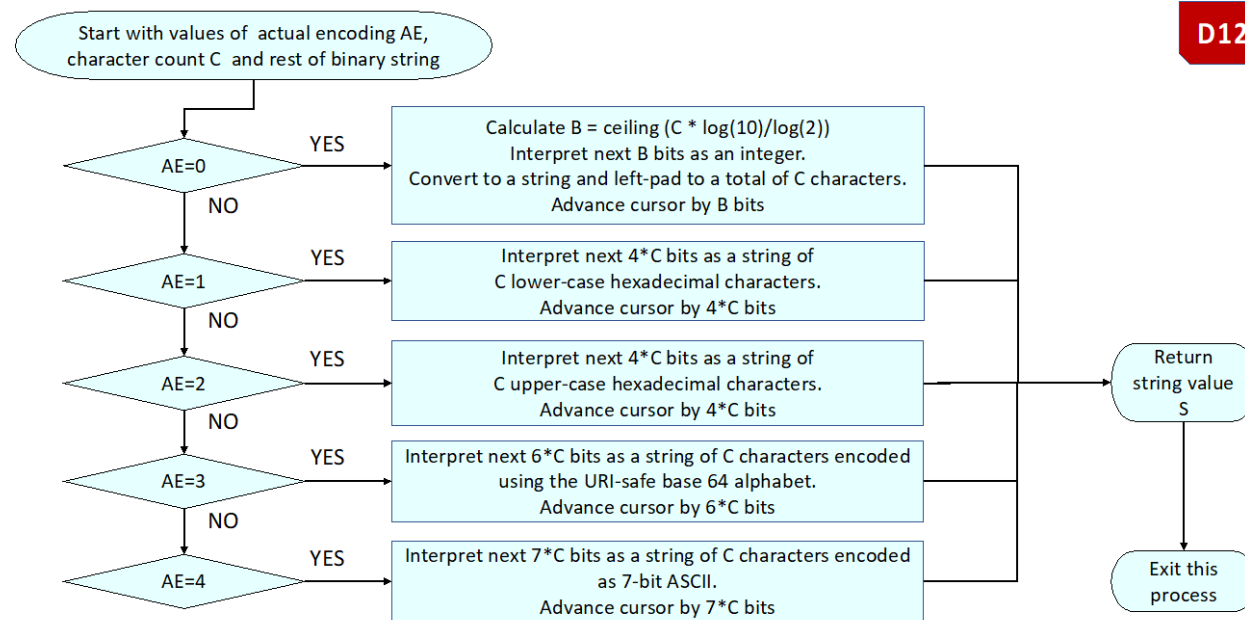
Figure 8-26 D10: Process as a single GS1 Application Identifier


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 8-27 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 8-28 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 8-29 D13: Extract the value for key=value pairs

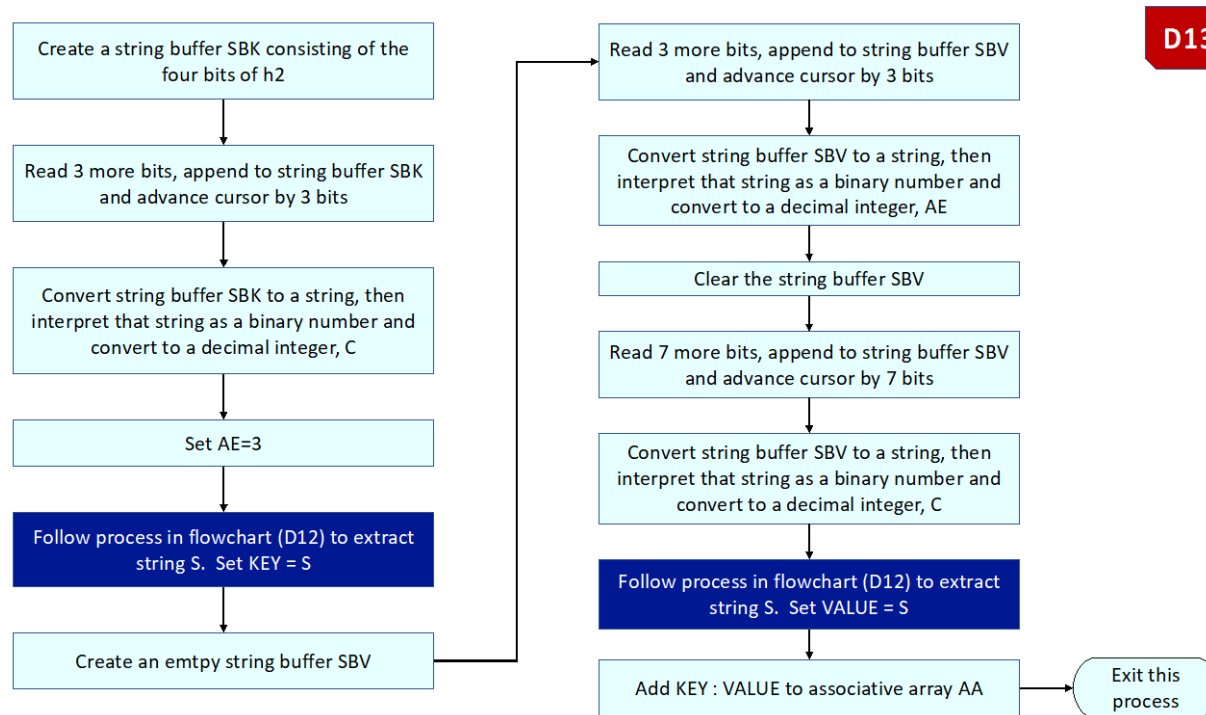


Figure D13 explains how to extract non-GS1 key:value pairs from a compressed string; after decompression these are restored to the URI query string. The four bits of h2 and a further 3 bits are interpreted as a 7-bit length indicator, permitting a key up to 127 characters. The key is always encoded using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

9 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.

9.1 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.

9.2 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 section [7.9](#) for more on relationship types.

9.3 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.

9.4 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.

9.5 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 [URIDoc] [HR14] 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'

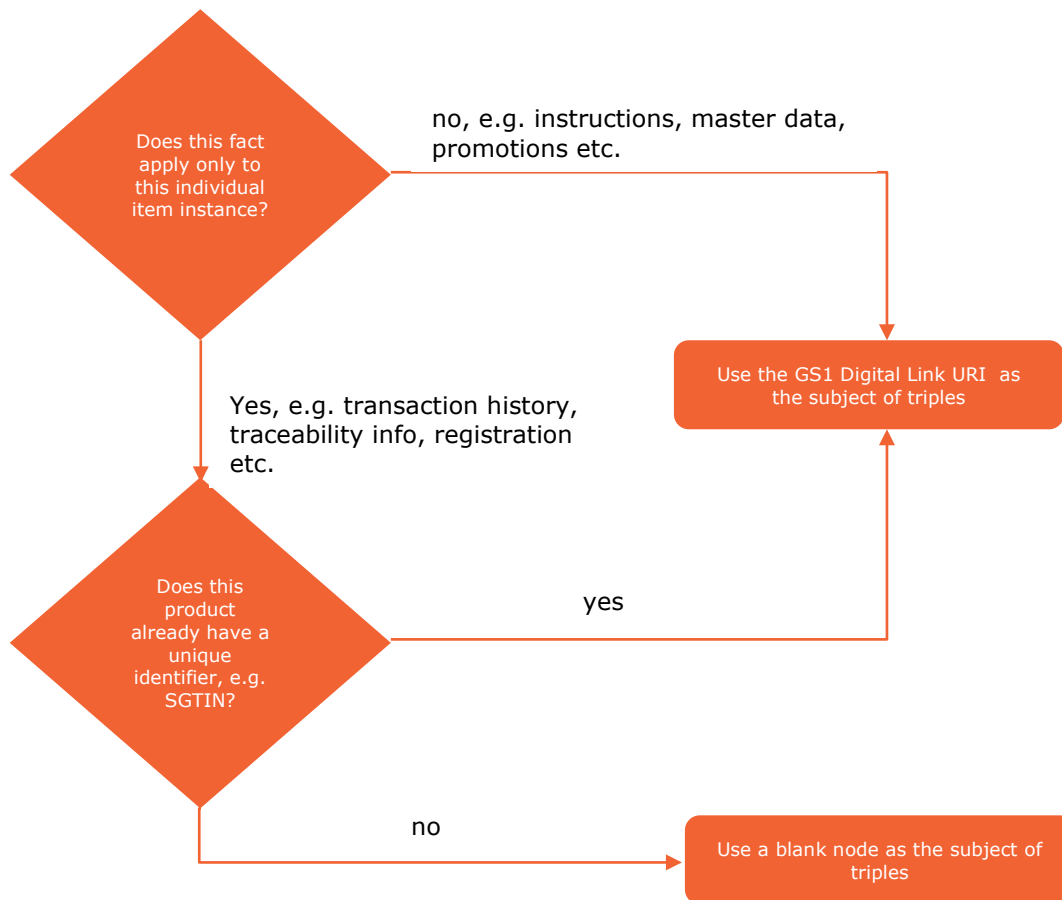
By using a blank node to make statements about 'this thing', we avoid making invalid statements that would otherwise apply too broadly to all instances of the same product.

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 9-1 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 9-1 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

9.6 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 [9.7.4](#)).

9.7 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 [GS1Voc] provides properties for all GS1 attribute AIs.

9.7.1 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.

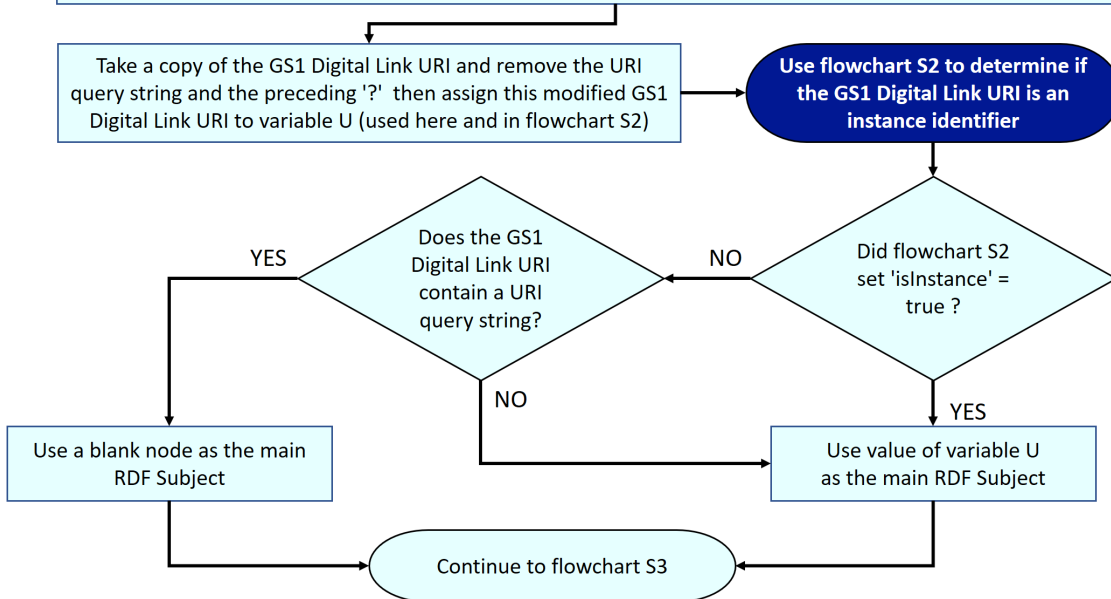
The third column indicates a 'minimum length'. For a GS1 primary identification key in table S1, if a minimum length is indicated in the third column and its value is equal or greater than the minimum length, a unique instance exists. This is to support GS1 Application Identifiers that have optional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

AI of GS1 Primary ID key	Requires additional AIs	Minimum Length
01	21, 235	
00	null	
253		14
254		14
8003		15

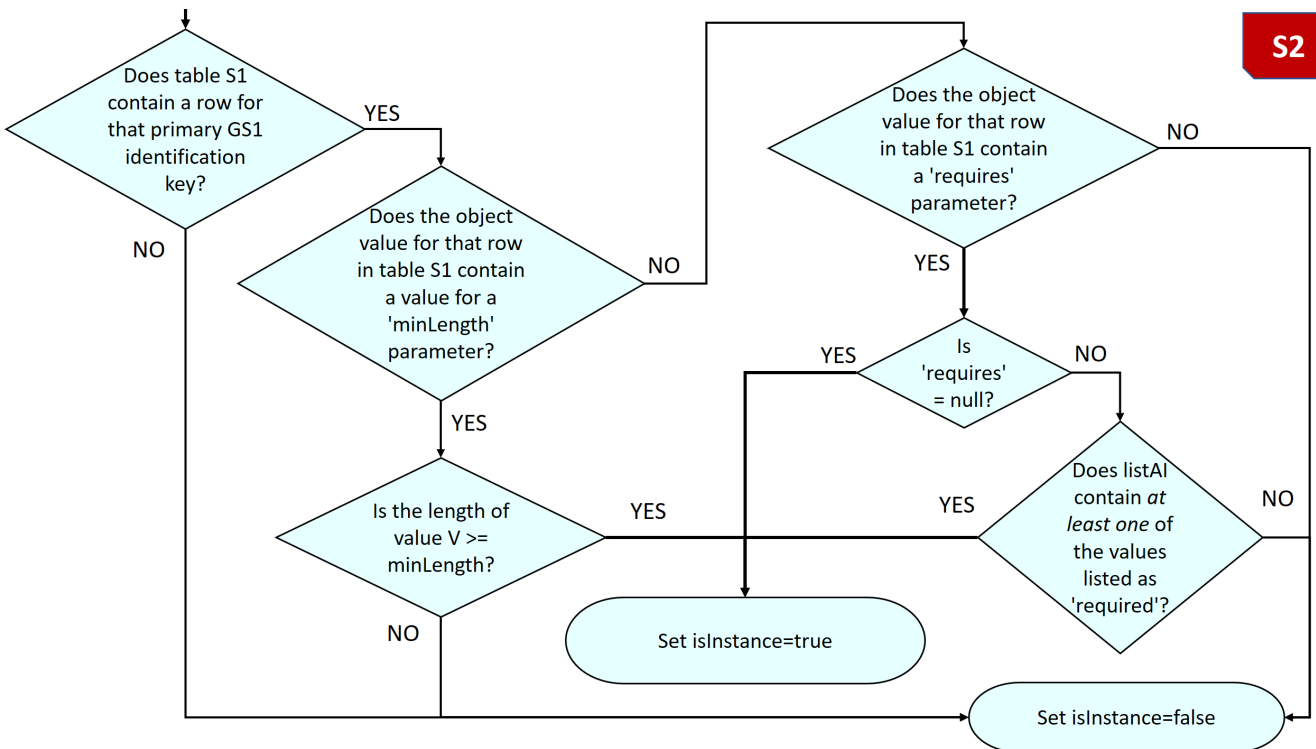
AI of GS1 Primary ID key	Requires additional AIs	Minimum Length
8004	null	
8006	21	
8010	8011	

S1

From the GS1 Digital Link URI extract the primary GS1 identification key as a numeric AI (e.g. '01') and extract the corresponding value V (e.g. '01234567890128'). Also extract a list of all other GS1 Application Identifiers expressed within that GS1 Digital Link URI, convert them to a list of numeric strings (e.g. '10', '21') and store in a list named listAI



S2



9.7.2 Determine class relationships that can be inferred

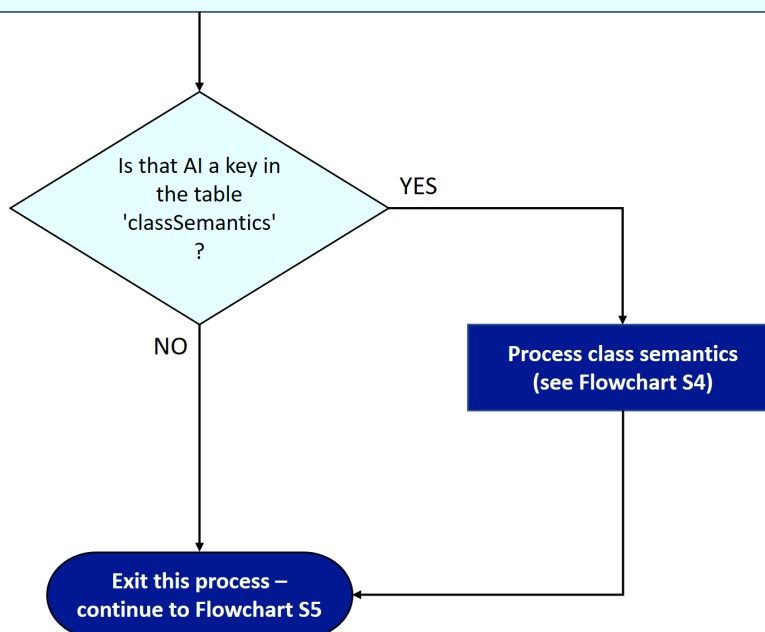
Table 'classSemantics' shows relationships between GS1 Application Identifiers of primary GS1 identification keys and corresponding semantic classes that can be inferred for such AIs.

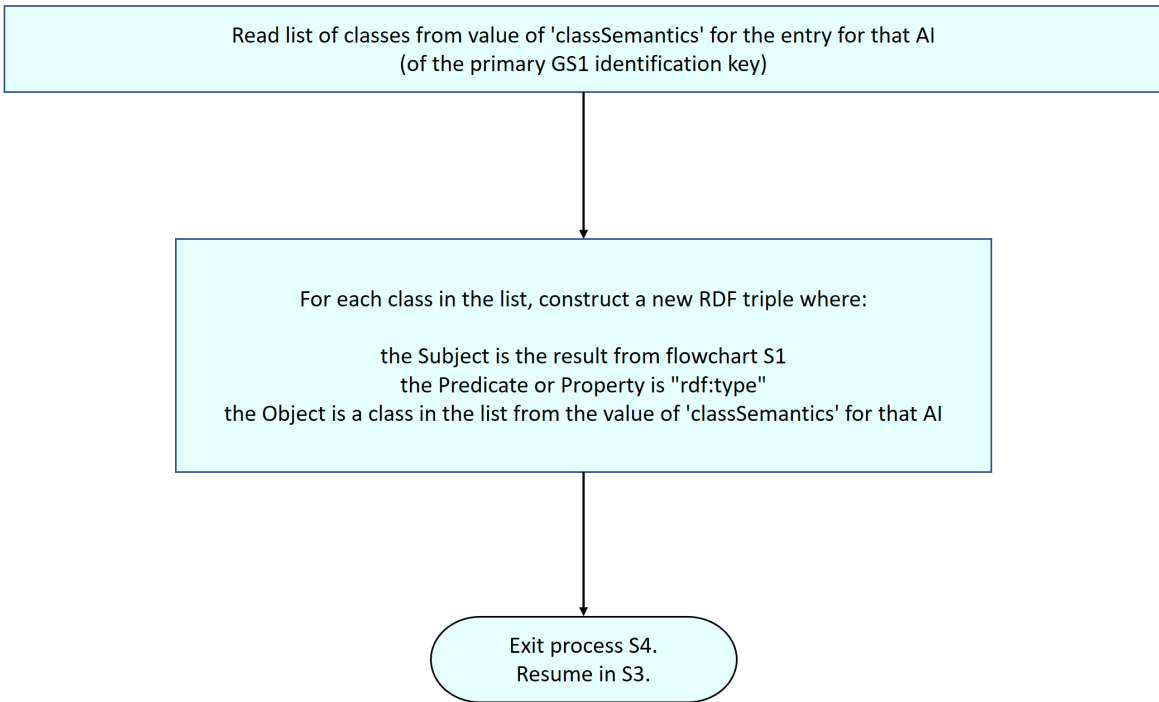
AI of GS1 Primary ID key	Inferred class
01	gs1:Product , schema:Product
8006	gs1:Product , schema:Product
414	gs1:Place , schema:Place
417	gs1:Organization, schema:Organization

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

S3

From the GS1 Digital Link URI extract the numeric GS1 Application Identifiers for the primary GS1 identification key within that GS1 Digital Link URI (e.g. '01')





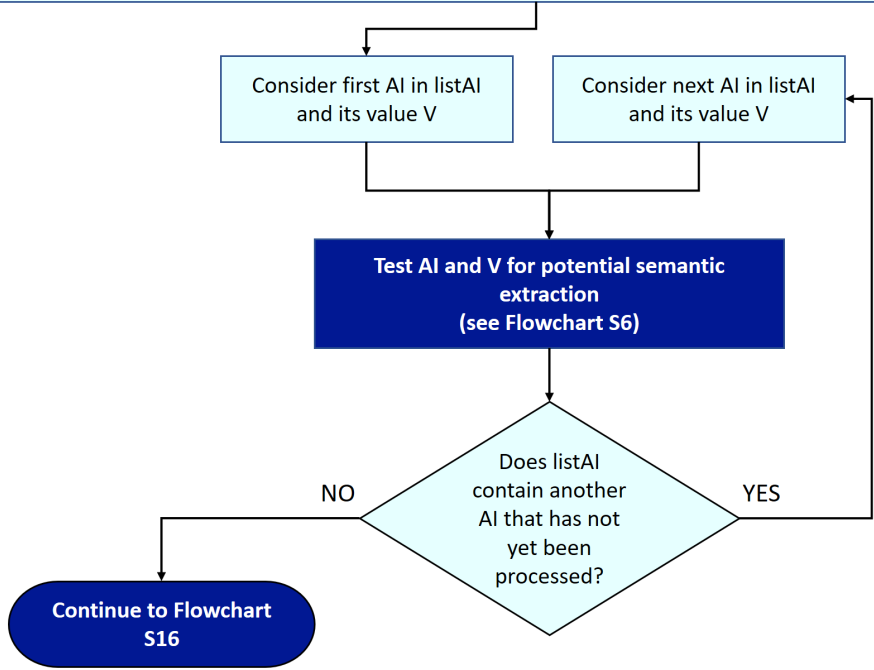
9.7.3 Extract further relationships expecting data values

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.

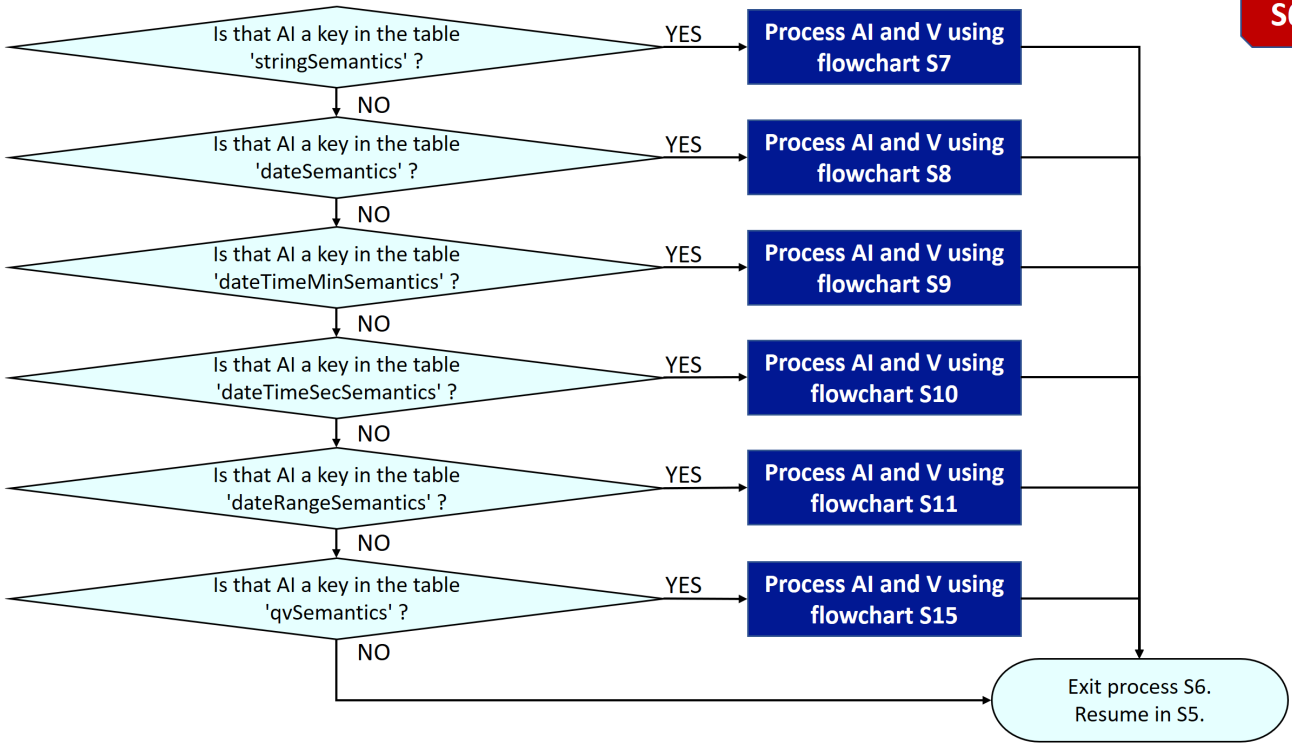
As shown in Flowchart S5, the first step is to convert a GS1 Digital Link URI into a table that maps each GS1 Application Identifier present within the GS1 Digital Link URI to its value, then use Flowchart S6 and its dependent flowcharts (S7 to S15) to perform appropriate reformatting of the value so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

S5

From the GS1 Digital Link URI extract a list of all GS1 Application Identifiers expressed within that GS1 Digital Link URI, convert them to a list of numeric strings (e.g. '10', '21') and store in a list named listAI



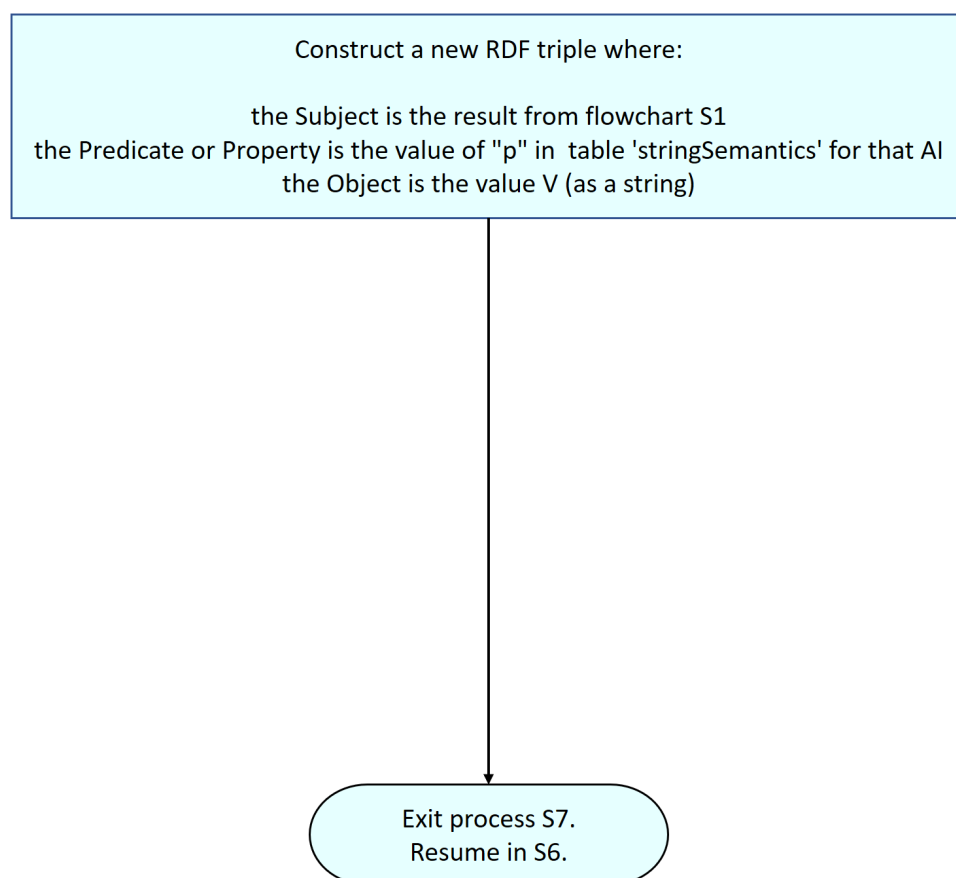
S6



9.7.3.1 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)

S7


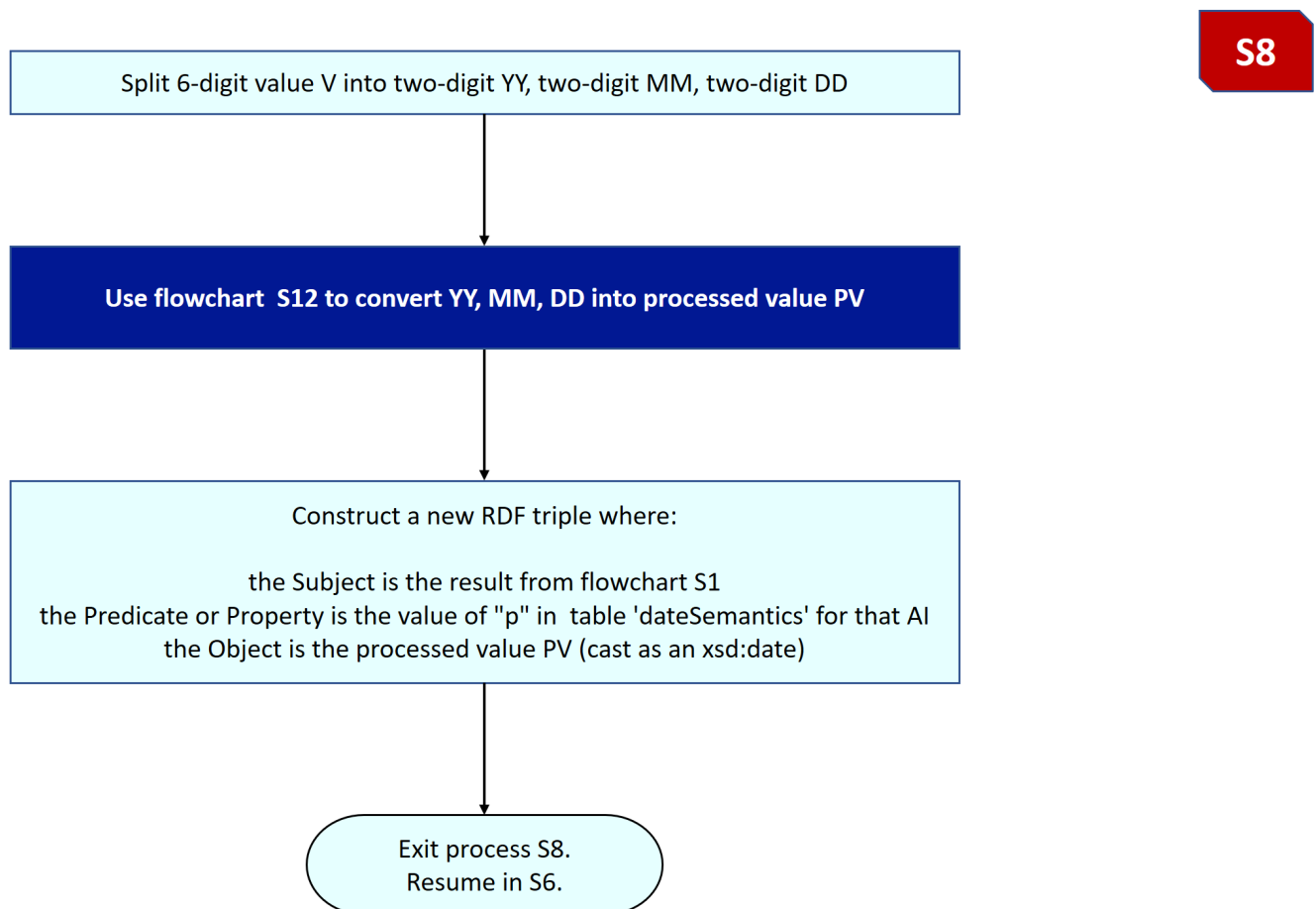
9.7.3.2 Properties expecting date values

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

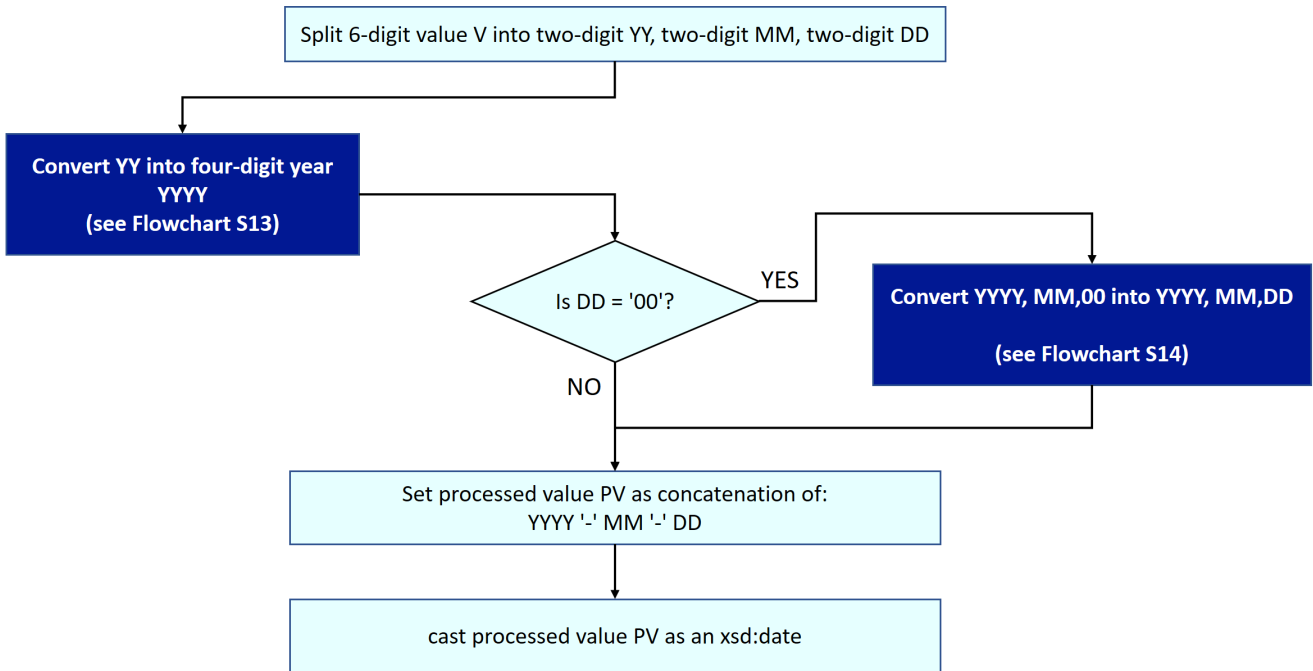
GS1 Application Identifier	Corresponding property	Explanation
11	gs1:productionDate	Date of Production
12	gs1:dueDate	Due date (for payment of invoice)
13	gs1:packagingDate	Packaging date

GS1 Application Identifier	Corresponding property	Explanation
15	gs1:bestBeforeDate	Best before date
16	gs1:sellByDate	Sell by date
17	gs1:expirationDate	Expiration date
7006	gs1:firstFreezeDate	First freeze date

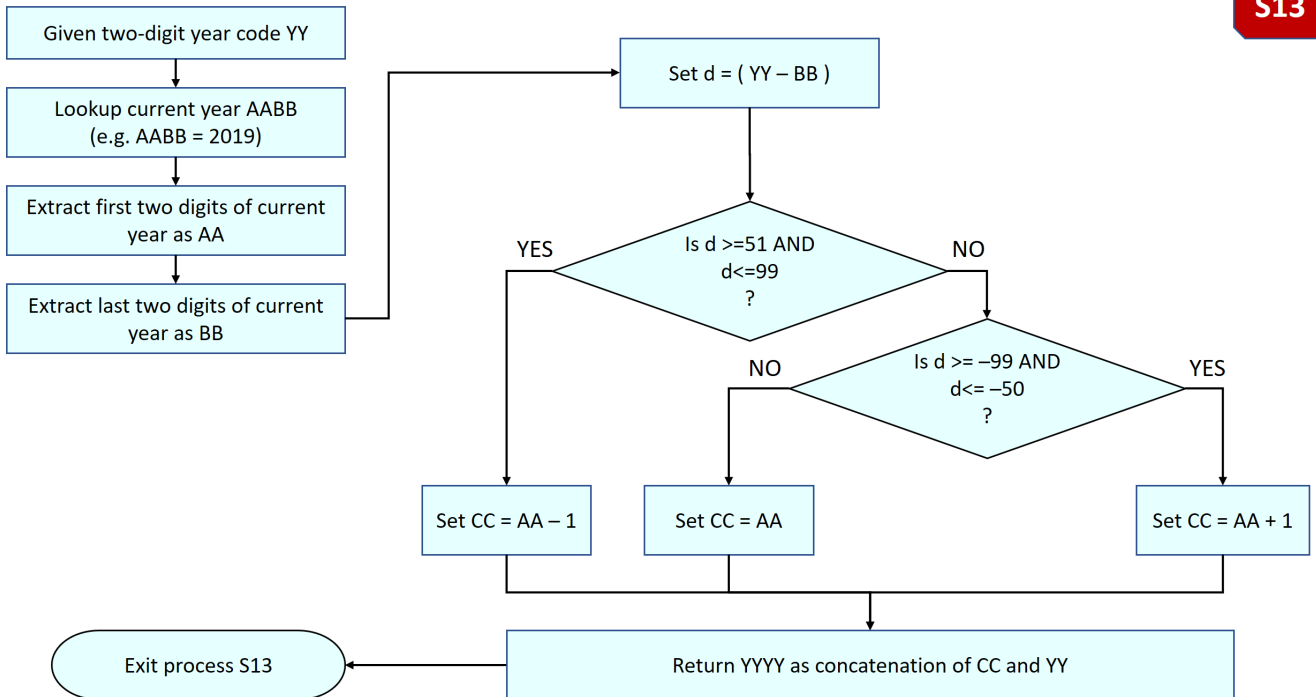
Flowcharts S8, S12, S13 and S14 explains the logic for reformatting a date from format YYMMDD into an xsd:date value. Flowchart S13 handles correct conversion of a two-digit year YY into a four-digit YYYY value. Flowchart S14 handles conversion of a two-digit date expressed as '00' to mean "last day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

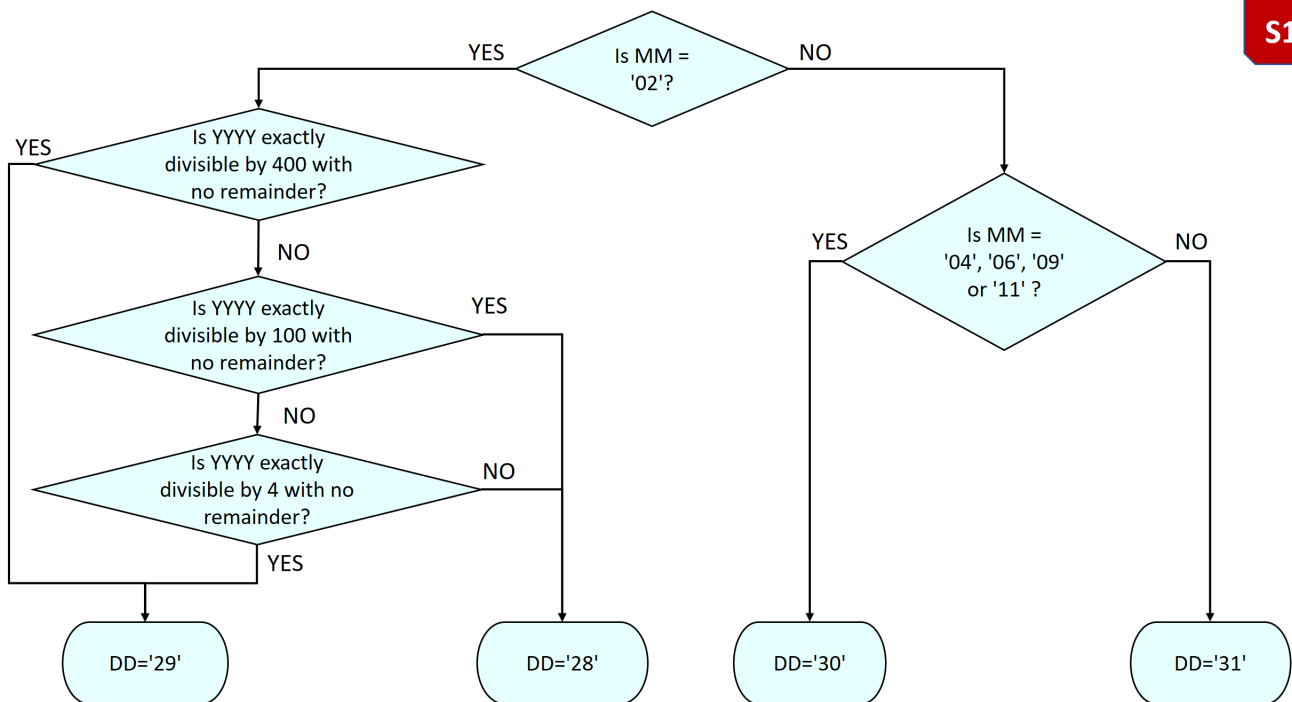


S12



S13

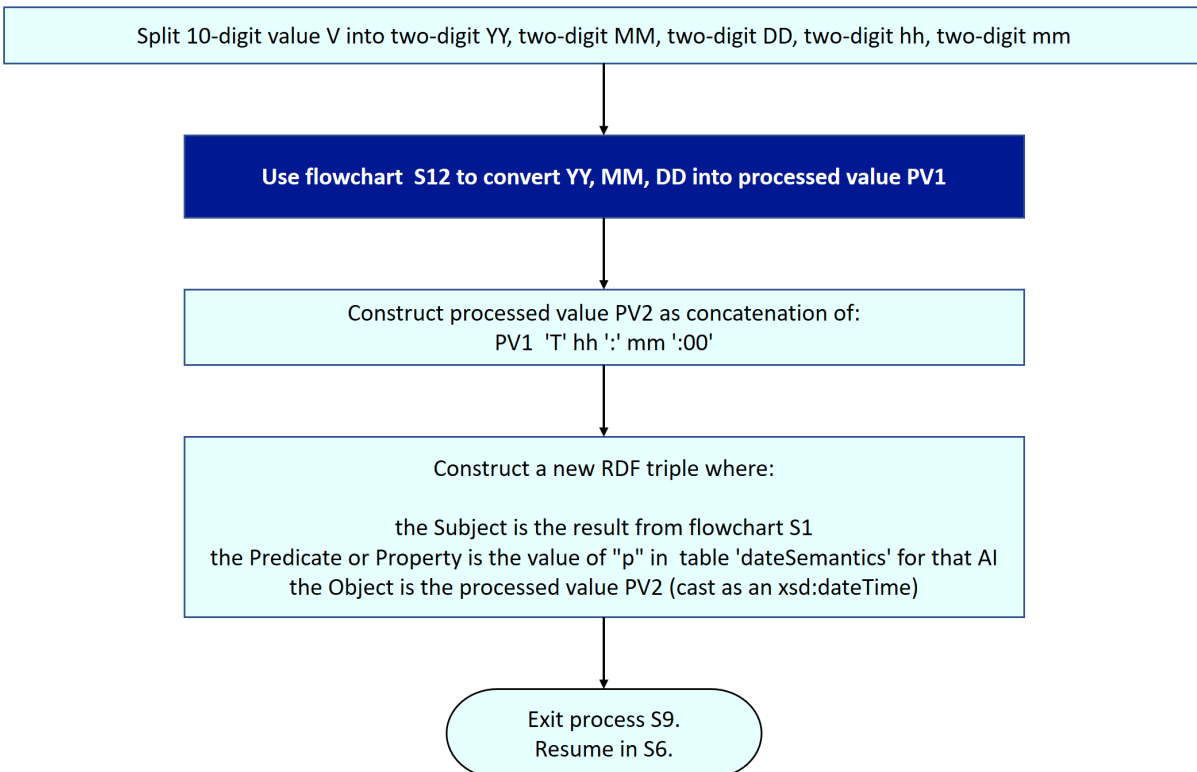




9.7.3.3 Properties expecting date`Time` values with minute granularity

Table 'date`Time`MinSemantics' 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.

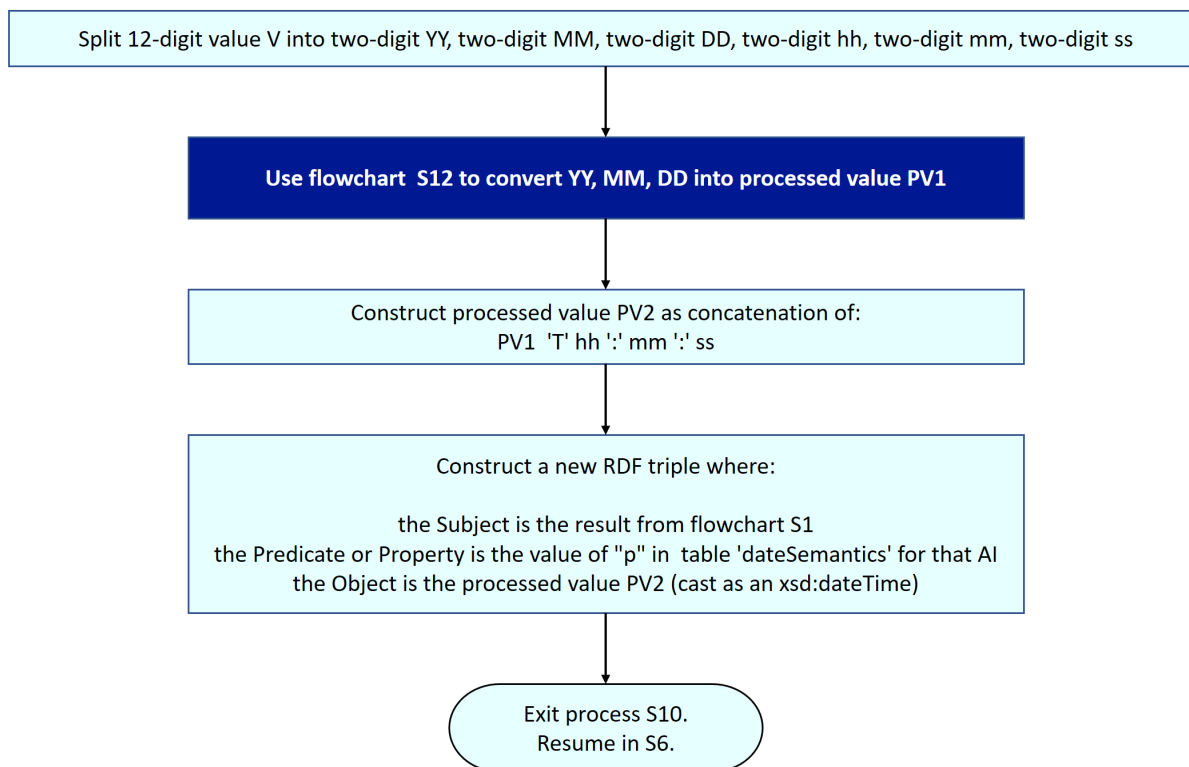
GS1 Application Identifier	Corresponding property	Explanation
7003	<code>gs1:expirationDate<code>Time</code></code>	Expiration date and time



9.7.3.4 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.

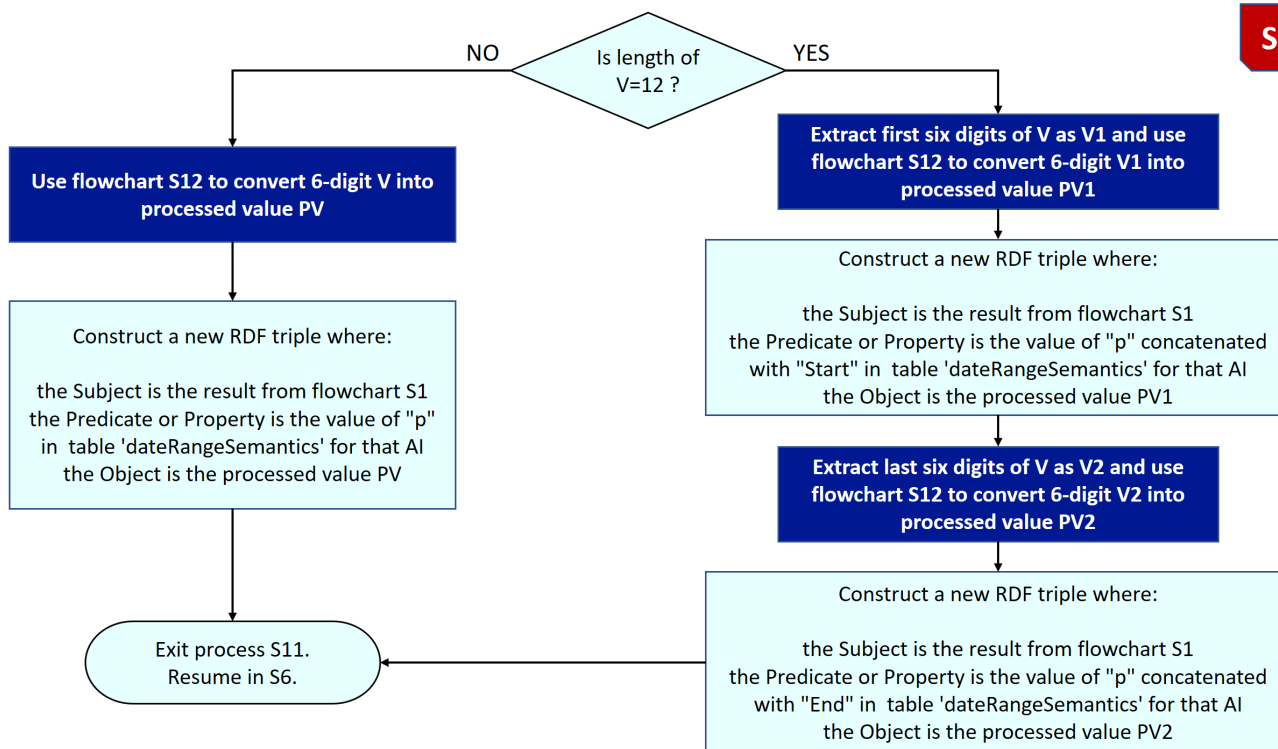
GS1 Application Identifier	Corresponding property	Explanation
8008	gs1:productionDateTime	Date and time of production



9.7.3.5 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	Corresponding property	Required matching format of value
7007	gs1: harvestDate	YYMMDD
	gs1: harvestDateStart	YYMMDDYYMMDD
	gs1: harvestDateEnd	YYMMDDYYMMDD



9.7.3.6 Properties expecting quantitative values (value and unit code)

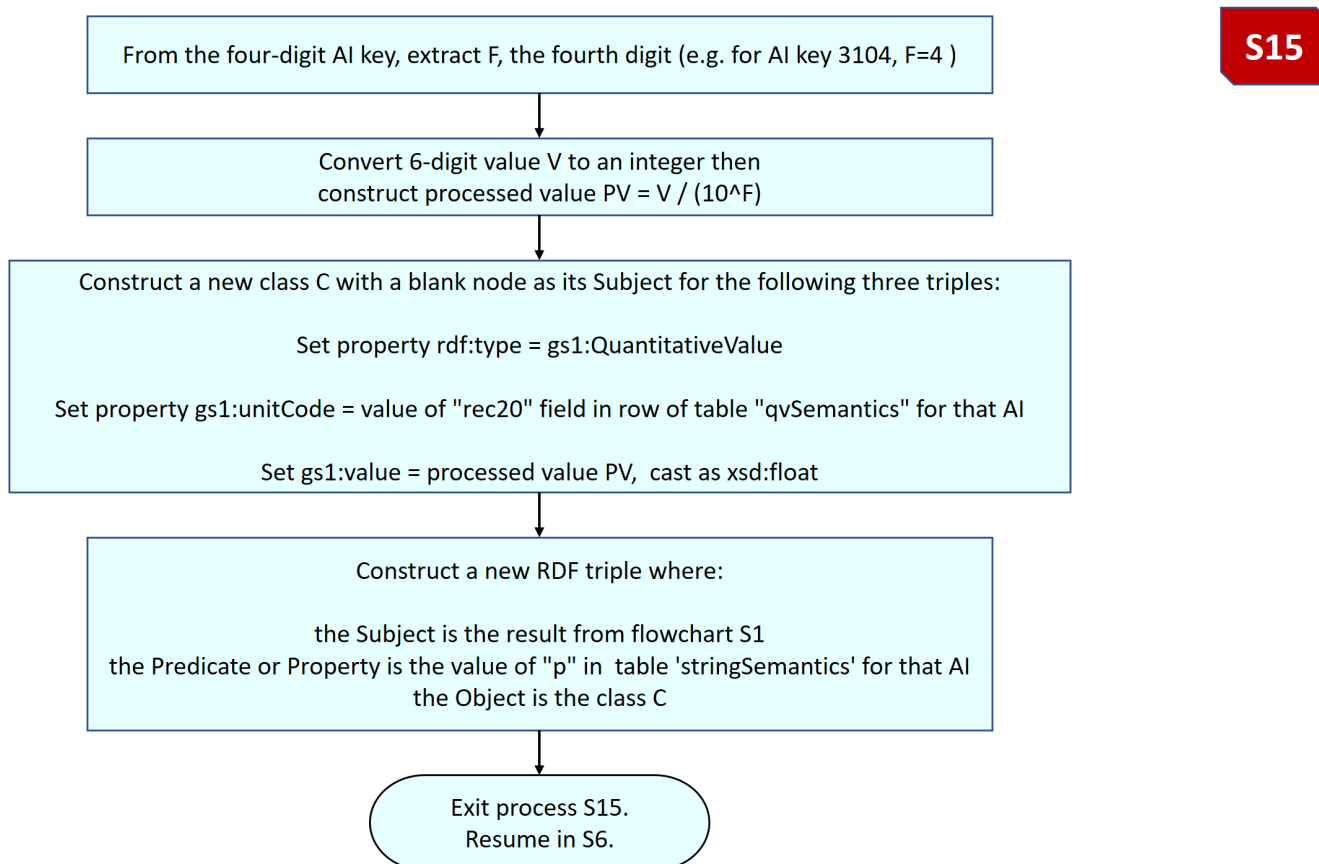
Table 'qvSemantics' below lists GS1 Application Identifiers whose values correspond to quantitative values (`gs1:QuantitativeValue`, `schema:QuantitativeValue`) that have a numeric floating-point value and an associated unit code, expressed using the appropriate UN ECE Recommendation 20 unit code. Note that unlike the previous tables, the first column of table 'qvSemantics' expresses a range of GS1 Application Identifiers. For example, considering the first row, $310n$ 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, $310n$, $320n$, $356n$, $357n$ 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
$310n$	<code>gs1:netWeight</code>	KGM
$320n$	<code>gs1:netWeight</code>	LBR
$356n$	<code>gs1:netWeight</code>	APZ
$357n$	<code>gs1:netWeight</code>	ONZ
$330n$	<code>gs1:grossWeight</code>	KGM
$340n$	<code>gs1:grossWeight</code>	LBR
$315n$	<code>gs1:netContent</code>	LTR
$316n$	<code>gs1:netContent</code>	MTQ

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

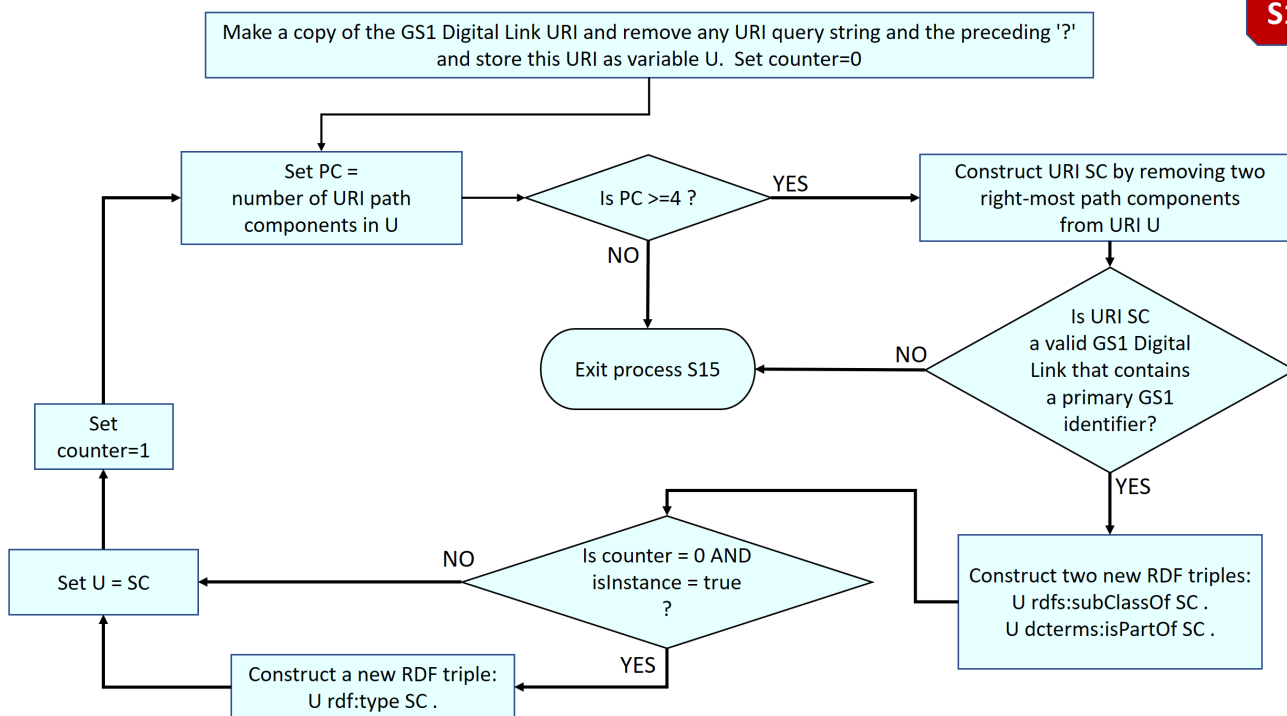
GS1 Application Identifier range (where $n = 0$ through 5)	Corresponding property	UN ECE Recommendation 20 unit code
350 n	gs1:netArea	INK
314 n	gs1:netArea	MTK
352 n	gs1:netArea	YDK
354 n	gs1:grossArea	FTK
353 n	gs1:grossArea	INK
334 n	gs1:grossArea	MTK
355 n	gs1:grossArea	YDK
337 n	gs1:massPerUnitArea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



9.7.4 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.



9.8 Worked examples

This subsection is informative

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

9.8.1 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 subclass 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.

9.8.2 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.
```

9.8.3 GTIN + Measured Weight

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

```
https://example.com/gtin/614141123452?3103=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 hve 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.
```

9.9 AIs defined as Web vocabulary terms

To support the semantic interpretation of GS1 Digital Link URIs, we propose that the 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

Proposed new property	GS1 Application Identifier	Range of new property
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	YYMMDDYYMMDD
gs1:harvestDateEnd	7007	xsd:date	YYMMDDYYMMDD
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

Proposed new property	GS1 Application Identifier range (n=0 to 5)	Range of new property
gs1:grossArea	334n 353n 354n 355n	gs1:QuantitativeValue
gs1:massPerUnitArea	337n	gs1:QuantitativeValue

9.10 Link type semantics

The majority of link types are defined in the GS1 namespace (<https://gs1.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.

10 Changes since version 1.0

Version 1.0 of the Digital Link standard was ratified and published in August 2018 [DL1]. The key differences between that document and this are listed below. Except in the minor instances described in the first two points below, the normative content from version 1 is included in this version and remains unchanged.

- The definition of a canonical GS1 Digital Link URI now states that all GTINs SHALL be expressed as 14 digits (with leading zeroes as necessary). This change is reflected in section [5.11](#) and many of the examples in section [6](#). Further rules added in this version mean that in a canonical DL URI, the query parameters are in a defined order so that they can be digitally signed.
- AIs 417, 235 and 7040 have been added to the normative definition of a GS1 Digital Link URI (section [5.2](#) with minor effects throughout the rest of section [3](#)).
- This version of the GS1 Digital Link standard provides normative definitions for resolvers (section [7](#)), a compression and decompression algorithm designed to maximise the length of GS1 Digital Link URIs that can be encoded in limited capacity data carriers (section [8](#)), and the semantics of a GS1 Digital Link URI (section [9](#)). Some of this content was present in version 1 of the standard as informative text, but has been extended and updated before being included here as normative sections of the specification.
- Much of the informative content of version 1.0 has been removed as it reflected ideas that have now been made more formal in the new normative sections, moved to other documents, or found to be unnecessary.
- Informative text that looked ahead to the definition of a resolver used the parameter 'typedLink'. This has been changed to `linkType` in version 1.1 (and in the Lightweight Messaging Service standard [LMS]).

- The semantics of a GS1 Digital Link URI have been made explicit.

10.1 Deferred to version 1.2

The working group notes that further work will be necessary beyond this version of the GS1 Digital Link standard. It is anticipated that the following issues will be tackled in the next iteration:

- A standard ingestion API for resolvers such that data containing links associated with one or more GS1 Digital Link URIs can be uploaded to multiple resolvers. This is likely to include the ability to use pattern matching (regular expressions) to associate links with sets of URIs.
- Methods to authenticate that links associated with a GS1 Digital Link URI are brand authorised.
- The format of the JSON object that represents the set of links associated with a given GS1 DL URI is likely to be standardised.
- Finalising of the semantics chapter (9), in particular, its use of schema.org
- There are some combinations of GS1 Application Identifiers that are not defined as invalid pairs in the GS1 General Specifications (see section 4.14.1 of GS1 Gen Specs v19.1) but which cannot be represented within a single GS1 Digital Link URI. For example, consider a physical object such as a tool or piece of electrical equipment that was identified by a GTIN (and possibly also a serial number) and which now serves as an individual asset e.g. within a school, hospital or other workplace and is now identified also by a GIAI. Section 4.14.1 of the GenSpecs does not forbid the object from carrying a GTIN and a GIAI but with the current ABNF grammar defined in section 5, it is not possible to express both the GIAI value and GTIN value within the same GS1 Digital Link URI; either the GTIN or the GIAI can be selected as the primary identification key but the other cannot be included as a data attribute within the URI query string.
- Considering whether resolvers should ever be required to parse an incoming query string, for example to check an expiry date.
- Review content of the Resolver Description File, noting that the example in section 7.11.2 does not contain all the data included in the live GS1 resolver

10.2 Changes since version 1.1.0

Notices added to the title page and the beginning of chapters 4-6 highlighting that they have been superseded by GS1 Digital Link standard: URI syntax.

Typographical errors fixed in sections [8.4.4.2](#) and [8.6.4.1](#)

Notices added to the title page and beginning of section 7 highlighting the it has been superseded by the GS1-Conformant resolver standard.

11 Glossary

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

Term	Definition
Arch	GS1 System Architecture, version 8.0, February 2019 https://www.gs1.org/docs/architecture/GS1_System_Architecture.pdf
Attribute	An element string that provides additional information about an entity identified with a GS1 identification key, such as batch number associated with a Global Trade Item Number (GTIN).
Brand Owner	The organisation that owns the specifications of a trade item, regardless of where and by whom it is manufactured. The brand owner is normally responsible for the management of the Global Trade Item Number (GTIN).
Canonical GS1 Digital Link URI	The definitive GS1 Digital Link URI for a given resource. See section Canonical GS1 Digital Link URIs

Term	Definition
Certified information agency (also known as an 'accreditation agency')	An organisation which certifies that a product meets specific criteria (e.g., Fairtrade, Kosher, Halal, Electrical Rating)
Consumer	Often considered as the "recipient" of the supply chain in the past, today's consumer is an active part of the supply chain and expects more data, with higher accuracy, and greater ease.
Consumer Product Variant (CPV)	An alphanumeric attribute of a GTIN assigned to a retail consumer trade item variant for its lifetime.
Content Negotiation	Content negotiation is a mechanism for offering information in different data formats and different languages. When a user agent (such as a browser) makes an HTTP request, it sends along some HTTP headers to indicate what data formats and language it prefers. The server then selects the best match from its file system or generates the desired content on demand, and sends it back to the client.
Data Field	A field that contains a GS1 identification key, an RCN, or attribute information
Data titles	Data titles are the abbreviated descriptions of element strings which are used to support manual interpretation of barcodes.
Dereferencing a URI	The use of an appropriate access mechanism (e.g. Web request) to perform an action on the URI's resource (e.g. to retrieve an information representation via HTTP GET or to send data to a resource via an HTTP POST operation). Dereferencing a URI is often considered synonymous with making a Web request or 'looking up' a URI on the Web.
Domain name	<p>A domain name is an identification string that defines a realm of administrative autonomy, authority or control within the Internet. Domain names are formed by the rules and procedures of the Domain Name System (DNS). Any name registered in the DNS is a domain name. Domain names are used in various networking contexts and application-specific naming and addressing purposes.</p> <p>Domain names provide a abstraction layer that separates a registered name for an organisation or activity from the actual internet addresses (IP addresses) that provide its associated information services such as its Website, its e-mail server etc. The system that connects the domain names with the corresponding IP addresses is the Domain Name System (DNS).</p>
Element string	The combination of a GS1 Application Identifier and GS1 Application Identifier data field.
GS1 Application identifier	The field of two or more digits at the beginning of an element string that uniquely defines its format and meaning.
GS1 Application identifier data field	The data used in a business application defined by one GS1 Application Identifier.
GS1 Barcode	A data carrier which encodes GS1 Application Identifier element strings.
GS1 Barcode using GS1 Application Identifiers	All GS1 endorsed barcode symbologies that can encode more than a GTIN namely GS1-128, GS1 DataMatrix, GS1 DataBar and Composite and GS1 QR Code.
GS1 Identification key	A unique identifier for a class of objects (e.g. a trade item) or an instance of an object (e.g. a logistic unit).
GS1 key qualifier	A key qualifier is an additional attribute that is designated for use as part of a compound key (e.g., GTIN + serial number is a compound key, with the serial number being a key qualifier for the GTIN)
GS1 Resolver	A Web server that is able to understand the GS1 Digital Link URI syntax
GS1 Digital Link URI	A Web URI conforming to the GS1 Digital Link URI syntax.
HR14	HttpRange14Webography. The chronology of a permathread, Sandro Hawke, W3C Wiki, 2003 https://www.w3.org/wiki/HttpRange14Webography
HTTP status codes	The status-code element is a three-digit integer code giving the result of the attempt to understand and satisfy the request.
Identification number	A numeric or alphanumeric field intended to enable the recognition of one entity versus another.

Term	Definition
LGTIN (GTIN + Lot/Batch)	A compound key formed from the combination of GTIN [AI (01)] and Batch/Lot Identifier [AI (10)]. LGTIN is defined as an EPC Class URN in the current GS1 Tag Data Standard (v1.11), sections 6.4.1 and 7.14, which describes the mapping between the EPC Class URN format for LGTIN and the corresponding element string.
Media Type (also known as MIME type or Content type)	A two-part string identifier that indicates a data format as a pair of type and subtype, e.g. image/jpeg, image/gif, image/png, text/html, text/rtf Media types are sometimes also referred to as MIME types (MIME is an acronym of Multipurpose Internet Mail Extensions) or Content Types (after the HTTP header that indicates the Media type)
Mobile scanning	An approach to giving consumers access to additional information or services about trade items through their mobile device. It is the ability to retrieve additional information about the trade item through mobile devices or in general between link a trade item with virtual information or services.
Parsing	The process of analysing the structure of a sentence or URI structure in order to extract relevant information from it. Note that within the context of EPC URN structures, parsing refers to the ability to extract structural components within the EPC structure, e.g. for the purpose of matching against EPC URN patterns.
QR Code®	A two-dimensional matrix symbology consisting of square modules arranged in a square pattern. The symbology is characterised by a unique finder pattern located at three corners of the symbol. QR Code® symbols are read by two-dimensional imaging scanners or vision systems
Reference GS1 Digital Link URI	A GS1 Digital Link URI that uses the id.gs1.org domain
Resolver	A resolver connects a GS1-identified item to one or more online resources that are directly related to it. The item may be identified at any level of granularity, and the resources may be either human or machine readable. Examples include product information pages, instruction manuals, patient leaflets and clinical data, product data, service APIs, marketing experiences and more.
Retailer	An organisation engaged in the sale and distribution of products to consumers. Also includes online retailers / e-tailers
SGTIN (Serialised GTIN)	A compound key formed from the combination of a GTIN [AI (01)] with Serial Number [AI (21)] which provides globally unique identification for every instance of a product. The term SGTIN appears in section 6.3.1 and 7.1 of the current GS1 Tag Data Standard, v1.11
Subdomain	A subdomain is a domain that is part of a main domain. Although example.com is a subdomain of the top-level domain (.com), we most often think of a subdomain as the part of the hostname that precedes the registered domain name. For example, the registered domain name gs1.org has one subdomain ('www') [as in www.gs1.org] that is used for its Website. It also has a subdomain ('id') [as in id.gs1.org] that is used for Web-based data services for GS1.
URI	Uniform Resource Identifier. A string of characters used to identify a resource. The resource may be an information resource such as a Web page or a thing in the real world, such as a physical object, person or location. URIs refer to the superset of Uniform Resource Names (URNs), Uniform Resource Locators (URLs) and Web URIs (which can function both as globally unambiguous names, while also behaving like URLs by enabling intuitive retrieval of related information via the Web).
URI fragment identifier	The fragment identifier component of a URI allows indirect identification of a secondary resource by reference to a primary resource and additional identifying information. The identified secondary resource may be some portion or subset of the primary resource, some view on representations of the primary resource, or some other resource defined or described by those representations. A fragment identifier component is indicated by the presence of an octothorpe / hash / number sign ("#") character and terminated by the end of the URI. A typical use of a URI fragment identifier is to provide a direct link to a specific section within a very long Web document such as https://www.w3.org/TR/dwbp/#DataIdentifiers

Term	Definition
URI path information	<p>A path consists of a sequence of path segments separated by a slash ("/") character. A path is always defined for a URI, though the defined path may be empty (zero length). The path component contains data, usually organized in hierarchical form, that, along with data in the non-hierarchical query component, serves to identify a resource within the scope of the URI's scheme and naming authority (if any). The path is terminated by the first question mark ("?") or number sign ("#") character, or by the end of the URI.</p>
URI query string	<p>The query component contains non-hierarchical data that, along with data in the path component, serves to identify a resource within the scope of the URI's scheme and naming authority (if any). The query component is indicated by the first question mark ("?") character and terminated by a number sign ("#") character or by the end of the URI.</p>
URIDoc	<p>Providing and Discovering URI Documentation, W3C TAG Finding (draft) 2 February 2002. Jonathan Rees https://www.w3.org/2001/tag/awwsw/issue57/latest/</p>
URL	<p>Uniform Resource Locator (URL), a specific type of URI colloquially known as Web address. A URL is a URI starting with http or https .</p>

12 References

ID	Name	Details
[Apache-conneg]	Apache Negotiation Algorithm	http://httpd.apache.org/docs/2.2/en/content-negotiation.html#algorithm
[ARCH]	GS1 System Architecture	v7, GS1, February 2018 https://www.gs1.org/docs/architecture/GS1_System_Architecture.pdf
[BCP47]	Tags for Identifying Languages	https://tools.ietf.org/html/bcp47
[BCP190]	URI Design and Ownership	https://tools.ietf.org/html/rfc7320
CORS	Cross-Origin Resource Sharing	W3C recommendation https://www.w3.org/TR/cors/ and WHATWG standard https://fetch.spec.whatwg.org/
[DC]	Dublin Core Metadata Initiative	http://dublincore.org/documents/dcmi-terms/
[DL1]	Digital Link version 1.0	Originally titled GS1 Web URI Structure https://www.gs1.org/standards/Digital-Link/1-0
[DOI]	Digital Object Identifier	ISO 26324:2012 Information and documentation – Digital object identifier system https://www.iso.org/standard/43506.html
[GS1 Identification Keys]	GS1 Identification Keys	https://www.gs1.org/standards/id-keys
[GENSPECS]	GS1 General Specifications	v19, January 2019 https://www.gs1.org/sites/default/files/docs/barcodes/GS1_General_Specifications.pdf
[GS1Voc]	The GS1 Web Vocabulary	https://www.gs1.org/voc/
[HTML LT]	HTML Living Standard	https://html.spec.whatwg.org/multipage/links.html
[HTTPcodes]	Hypertext Transfer Protocol [RFC 2616], Section 10: Status Code Definitions	https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
HTTPS	HTTP Over TLS	https://tools.ietf.org/html/rfc2818
[IANA LR]	Link relations	https://www.iana.org/assignments/link-relations/link-relations.xhtml
[IANA MT]	Media Types	https://www.iana.org/assignments/media-types/media-types.xhtml
[IRIs]	Internationalized Resource Identifiers (IRIs)	https://tools.ietf.org/html/rfc3987
[JSON-LD]	A JSON-based serialization for Linked Data	https://www.w3.org/TR/json-ld/
[Linked Data]	Linked Data, Tim Berners-Lee, 2006	https://www.w3.org/DesignIssues/LinkedData

ID	Name	Details
[Linkset]	Linkset: Media Types and a Link Relation Type for Link Sets draft-wilde-linkset-03	https://tools.ietf.org/html/draft-wilde-linkset-03
[LMS]	GS1 Lightweight Messaging Standard for Verification of Product Identifiers	https://www.gs1.org/verification-messaging
[PercentEncoding]	Uniform Resource Identifier (URI): Generic Syntax, section 2.1: Percent-Encoding	https://tools.ietf.org/html/rfc3986#section-2.1
[RDF]	Resource Description Framework Primer	https://www.w3.org/TR/rdf11-primer/
[RDF Intro]	Introduction to RDF Metadata	https://www.w3.org/TR/NOTE-rdf-simple-intro
[RFC 2606]	Reserved Top Level Domain Names	https://tools.ietf.org/html/rfc2606
[RFC 3986]	Uniform Resource Identifier (URI): Generic Syntax	https://tools.ietf.org/html/rfc3986
[RFC4648]	The Base16, Base32, and Base64 Data Encodings	https://www.rfc-editor.org/rfc/rfc4648.txt
[RFC 5234]	Augmented BNF for Syntax Specifications: ABNF	https://tools.ietf.org/html/rfc5234
[RFC8615]	Well-Known Uniform Resource Identifiers (URIs)	https://tools.ietf.org/html/rfc8615
[RFC 6570]	URI Template	https://tools.ietf.org/html/rfc6570
[RFC 6596]	The Canonical Link Relation	https://tools.ietf.org/html/rfc6596
[RFC 6761]	Special-Use Domain Names	https://tools.ietf.org/html/rfc6761
[RFC 7231]	HTTP/1.1 Semantics and Content, section 3.4, "Content Negotiation"	https://tools.ietf.org/html/rfc7231#section-3.4

ID	Name	Details
[RFC 7235]	HTTP/1.1 Authentication	https://tools.ietf.org/html/rfc7235
[RFC 7405]	Case-Sensitive String Support in ABNF	https://tools.ietf.org/html/rfc7405
[RFC8187]	Indicating Character Encoding and Language for HTTP Header Field Parameters	https://tools.ietf.org/html/rfc8187
[RFC8288]	Web Linking	https://tools.ietf.org/html/rfc8288
SKOS	SKOS Simple Knowledge Organization System	W3C Recommendation 18 August 2009 https://www.w3.org/TR/skos-reference/

A.1 Intellectual Property

A.1.1 Introduction and Disclaimer

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

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

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

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

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

A.1.2 Notices

A.1.2.1 US Patent No. US 9,794,321B2

EVERYTHNG Limited of 122 East Road, London N16FB (United Kingdom), the owner of the US Patent No. 9,794,321 (for the purpose of this specific paragraph A.1.2.1, the “**Patent**”), who participated to the Work Group designing the GS1 Digital Link Standard Release 1.1, gave notice that they believe the Patent contains Essential Claims for the implementation of GS1 Digital Link Standard Release 1.1 for the following sections:

- 7.1 Resolver Functionality
- 7.6 Associating links with identified items
- 7.7.2.1 Granular identifiers
- 7.7.2.2 Examples
- 7.7.3 Requesting a specific link type (client side)
- 7.7.5 The context keyword
- 7.7.6 Recognising the user’s language and requested media type

The owners have offered to grant a royalty free, non-exclusive license to the Patent to users of the GS1 Digital Link Standard Release 1.1, for use with their application of resolvers, and subject to (i) GS1 including a notice of the availability of the license in its documentation of the Standard, (ii) users who wish to benefit of the license registering with the owner to receive the license (via <https://evrythng.com/gs1-license/>).

The term of the offered license will be for the life of the Patent. For the avoidance of any doubt, this licensing offer pertains only to implementations of Release 1.1 of the GS1 Digital Link Standard.

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 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.1.2.2 Patents and Patent Applications of Mobilead SAS

MobiLead SAS of 1 Cour du Havre, 75008 Paris (France), the owner of the granted patent and five patent applications listed in this sub-section (for the purpose of this specific paragraph A.1.2.2, 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:

- 8.4 Structure of the compressed string
- 8.7 Formal ABNF grammar for compressed GS1 Digital Link URIs
- 8.8 Compression procedure and flowcharts
- 8.9 Decompression procedure and flowcharts

For the patent applications US 20180025195, EP 3276503 & CN 107657291, relevance is claimed for the following sections:

- 7.4 Link metadata
- 7.9 Redirection
- 9.3 Information encoded within the URI

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.1.2.3 Patents and Patent Applications from Servicetag SAS

Servicetag SAS of 171 bis, Avenue Charles de Gaulle, 92200 Neuilly-sur-Seine (France), the applicant of the patent application EP 18306189 (for the purpose of this specific paragraph A.1.2.3, 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:

- 8. Compression and decompression
- 7.1. (1.) Resolver Functionality (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.