GS1 Trusted Source of Data (TSD) Standard

the normative specification for standard data and interfaces in the Trusted Source of Data (TSD) framework.

Release 1.2, Ratified, May 2015

Document Summary

Document Item	Current Value
Document Name	GS1 Trusted Source of Data (TSD) Standard
Document Date	May 2015
Document Version	1.2
Document Issue	
Document Status	Ratified
Document Description	the normative specification for standard data and interfaces in the Trusted Source of Data (TSD) framework.

Contributors

Name	Organisation
Marc Benhaim, working group co-chair	GS1 France
Edward Collins, working group co-chair	Brandbank
Steven Robba, working group co-chair	1WorldSync Holdings, Inc.
Alvarez, Pete	GS1 GO
Angelo, Kerry	GS1 Community Room Staff
Ausili, Andrea	GS1 Italy
Beideman, Robert	GS1 GO
Beno, Martin	GS1 Slovakia
Bersani, Bob	GS1 Global Office
Besford, Robert	GS1 UK
Bigler, Lori	The J.M. Smucker Company
Bixler, Jeffrey	Georgia Pacific
Bradley, Ardetha	Georgia Pacific
Brown, Scott	GS1 US
Burd, Randy	MultiAd Kwikee
Dean, Kevin	GS1 Canada
Dicks, Arne	GS1 Germany
Edison, Carol	GS1 US
el kalla, ahmed	GS1 Egypt
Espinosa, Juliet	GS1 Columbia
Fahoum, Terri	The J.M. Smucker Company
Felipe, Juan	GS1 Columbia
Gathmann, Stefan	GS1 Ireland
Gerasimenko, alexander	Mars, Inc.
Gray, Neil	GS1 UK
Green, Cameron	GS1 Global Office
Gupta, Sudu	ITradeNetwork.com, Inc.

Name	Organisation
Hakala, Pertti	GS1 Finland
Herregodts, Kurt	GS1 Belgium & Luxembourg
Hyler, Alan	GS1 Global Office
Ichihara, Hideki	GS1 Japan
Iwasaki, Yoshihiko	GS1 Japan
Kauz, Eric	GS1 Global Office
Kidd, Robin	Nestle
Kolwane, Neville	GS1 South Africa
Kovacic, Bojan	GS1 Slovenia
Kurmanova, Kamilla	GS1 Russia
Lekwana, Pedro	GS1 South Africa
Lockhead, Sean	GS1 Global Office
Meldgaard, Charlotte	GS1 Denmark
Moreno, Sandra	GS1 Colombia
Moreno, Sandra	GS1 Columbia
Moritz, Marcus	GS1 Germany
Nachman-Ghnassia, Sophie	France Telecom Orange
Nuce, Melanie	GS1 US
Nunez, Katrin	Summa Technology Group
Ochoa, Juan	GS1 Colombia
Ogandzhanov, Georgy	GS1 Russia
Ottiker, Michel	GS1 Switzerland
Paixac, Silverio	GS1 Portugal
Pereira, Steven	GS1 Australia
Picoito, Joao	GS1 Portugal
Pujol, Xavier	GS1 Spain
Ramos, Carlos	GS1 Mexico
Rasmussen, Thomas	GS1 Denmark
Reithmeier, Michaela	GS1 Austria
Repec, Craig Alan	GS1 Global Office
Richardson, Rich	GS1 US
Roberts, Toni	Costco
Rubio Alegren, Sylvia	ICA AB
Sarachman, Michael	GS1 Global Office
Schins, Armand	Ahold (Europe)
Scott, Luke	Unknown-On Webex
Sieira, Marcel	GS1 Australia
Simonalle, Kim	epcSolutions inc.
Sinitsyn, Andrey	ItRuStore Ltd.
Slone, Joe	1WorldSync Holdings, Inc.

Name	Organisation
Soboleva, Olga	GS1 Russia
Spindler, Mike	ShelfSnap LLC
Stein, Sylvia	GS1 Netherlands
Tomassi, Gina	PepsiCo, Inc.
Traub, Ken	Ken Traub Consulting LLC
Vacval, Milan	Gladson Interactive
Whittington, Wade	Georgia Pacific LLC
Zegre, Michael	Saphety Level SA
Dipan Anarkat, B2C working group facilitator	GS1 Global Office
Mark Frey, GSMP working group facilitator	GS1 Global Office
Coen Janssen, Editor and modeller	GS1 Global Office
Ewa Iwicka, XML development	GS1 Global Office

Log of Changes

Release	Date of Change	Changed By	Summary of Change
i1.1	28-Mar-2013	Coen Janssen	▪ TSD_ProductData (paragraph 6.3): Added TSD_ProductComponentData class. ▪ Paragraph 6.4. Removed module definitions, these are now documented in [TSDPDM]. ▪ Added new paragraph 6.2.9 Formatted Description Modified paragraph 6.2.6 to support Product Components
i1.1-d5	17-Apr-2013	Coen Janssen	Rewrite of section 6.2.9 based on group decision of 17-Apr on formatted description.
i1.1-crd2	31-May-2013	Coen Janssen Ewa Iwicka	Comment resolutions
i1.1-crd3	4-Jun-2013	Coen Janssen Ewa Iwicka	Version for eBallot, including comment resolutions
i1.1	Jul-2013	Coen Janssen	eBallot successful, changed document status into Unratified and removed revision markings.
d1.2	Mar-2015	Mark Van Eeghem,	▪ Log of Changes added here ▪ Replaced all occurences of [ONS_20] by [ONS_201] based on comment by Dipan Anarkat. ▪ In the reference section on page 9, ONS is listed as a draft standard (not published). Changed that to Ratified standards and added the following link: http://www.gs1.org/gsmp/kc/epcglobal/ons/ons_2_0_1-standard-20130131.pdf Changed in same section Object Naming Service to Object Name Service.
1.2	May 2015	D. Buckley	Applied new GS1 Branding and release number following board ratification. No text changes whatsoever.

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 Trusted Source of Data (TSD) 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 organization 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, NONINFRINGMENT, FITNESS FOR PARTICULAR PURPOSE, OR ANY WARRANTY OTHER WISE ARISING OUT OF THIS SPECIFICATION. GS1 disclaims all liability for any damages arising from use or misuse of this Standard, 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

Introduction

This document is a GS1 Standard which is the normative specification for standard data and interfaces in the Trusted Source of Data (TSD) framework. The aim of the TSD framework to support the communication of authentic and accurate product data by brand owners to consumers/shoppers, retailers, internet applications, and government using internet and mobile devices. This standard defines the data and interfaces by which one or more Data Aggregators (as defined herein) may federate to provide seamless access to product data, for the benefit of Internet Applications across the globe.

1 Scope

This standard defines the following:

- The architectural framework within which the roles of Internet Application, Data Aggregator, and Global Index are defined. (See Sections 3 and 4.)
- Abstract data definitions for product data delivered by Data Aggregators to Internet Applications.
- Abstract interface definitions for message exchange, including:
  - Aggregator-Aggregator Query Interface (AAQI) One Data Aggregator querying another for data about a specified product (or products)
  - Aggregator-Index Query Interface (AIQI) A Data Aggregator querying the Global Index to obtain references to other Data Aggregators who may have data about a specified product (or products)
  - Aggregator-Index Maintenance Interface (AIMI) A Data Aggregator interacting with the Global Index to add, change, correct, or delete index entries that reference the Data Aggregator
- XML schemas that implement the abstract data definitions and the messages implied by the abstract interface definitions
- Bindings of the abstract interface definitions to specific transport protocols and associated security mechanisms, specifically:
  - A web services binding for the Aggregator-Aggregator Query Interface (AAQI). This binding makes use of the XML schemas for its data payloads.
  - A binding for the Aggregator-Index Query Interface (AIQI) based on ONS 2.01 [ONS_201]

This version of the standard does not define bindings for the AIMI.

Specifically out of scope of this standard is the interface between Internet Applications and Data Aggregators.

2 References

Normative references:

- [GS1PDM] GS1, GS1 Source Product Data Modules version 1.1, 2013
- [GS1GS] GS1, "GS1 General Specifications Version 12," January 2012.
- [ISO3166] "Codes for the representation of names of countries and their subdivisions – Part 1: Country codes," ISO 3166-1:2006.
- [ISO639] "Codes for the representation of names of languages – Part 1: Alpha-2 code," ISO 639-1:2002.
- [ISODir2] ISO/IEC Directives part 2; Rules for the structure and drafting of International Standards – 6th edition, 2011
- [ONS_201] GS1 Object Name Service 2.0, ratified standard, http://www.gs1.org/gsmp/kc/epcglobal/ons/ons_2_0_1-standard-20130131.pdf

- [RFC2246] T. Dierks, C. Allen, "The TLS Protocol, Version 1.0," RFC2246, January 1999, http://www.ietf.org/rfc/rfc2246.
- [RFC2818] E. Escorla, "HTTP Over TLS," RFC2818, May 2000, http://www.ietf.org/rfc/rfc2818.
- [RFC3268] P. Chown, "Advanced Encryption Standard (AES) Cipersuites for Transport Layer Security (TLS)," RFC3268, June 2002, http://www.ietf.org/rfc/rfc3268.
- [RFC3986] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC3986, January 2005, http://www.ietf.org/rfc/rfc3986.
- [UNECE20] United Nations Economic Commission for Europe, "Recommendation No. 20: Codes for Units of Measure Used in International Trade, Revision 7," September 2010, http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_Rev7e_2010.zip.
- [HTML 4.01] HTML 4.01 Specification, W3C Recommendation 24 December 1999

3 Terms and definitions

Within this specification, the terms SHALL, SHALL NOT, SHOULD, SHOULD NOT, MAY, NEED NOT, CAN, and CANNOT are to be interpreted as specified in Annex G of the ISO/IEC Directives, Part 2, 2001, 4th edition [ISODir2]. When used in this way, these terms will always be shown in ALL CAPS; when these words appear in ordinary typeface they are intended to have their ordinary English meaning.

All sections of this document, with the exception of the introduction, are normative, except where explicitly noted as non-normative.

The following typographical conventions are used throughout the document:

- ALL CAPS type is used for the special terms from [ISODir2] enumerated above.
- Monospace type is used to denote programming language, UML, and XML identifiers, as well as for the text of XML documents.
- Placeholders for changes that need to be made to this document prior to its reaching the final stage of approved GS1 specification are prefixed by a rightward-facing arrowhead, as this paragraph is.

For the purposes of this document, the following terms and definitions apply.

3.1 General Business Terms

Trade Item

In the context of this standard a Trade Item is defined as a product available for sale at a retail outlet or at an online store.

3.2 Architecture Terms for This Standard

Aggregator-Index Maintenance Interface (AIMI)

The interface through which a Data Aggregator interacts with the Global Index to add, change, correct, or delete index entries that reference the Data Aggregator.

Aggregator-Aggregator Query Interface (AAQI)

The interface through which one Data Aggregator queries another for data about a specified product (or products).

Aggregator-Index Query Interface (AIQI)

The interface through which a Data Aggregator queries the Global Index to obtain references to other Data Aggregators who may have data about a specified product (or products).

Bindings

The part of this standard that specifies concrete realisations of the Product Data Layer and the Service Interface Layer.

Data Aggregator

A service that gathers or maintains data regarding products, and makes this data available to Internet Applications. Each Data Aggregator is not expected to have data for all products, but instead interacts with peer Data Aggregators as necessary to create a federated system.

Data Pool

A data pool in the GS1 Global Data Synchronisation Network. A Data Pool is one source of data that a Data Aggregator draws from.

Global Index

A service that a Data Aggregator uses to identify peer Data Aggregators who have data about a given product, in the case where the first Data Aggregator does not itself have data about that product.

Internet Application

An application or service that is available to consumers via an Internet-based platform (desktop or mobile)

Product Data Layer

The part of this standard that specifies the data used to describe a Trade Item in the context of the TSD framework.

Service Interface Layer

The part of this standard that defines the three service interfaces: the Aggregator-Aggregator Query Interface (AAQI), the Aggregator-Index Query Interface (AIQI), and the Aggregator-Index Maintenance Interface (AIMI).

4 Business Background (non-normative)

The aim of the GS1 Business-to-Consumer Trusted Source of Data (TSD) framework is to support the communication of authentic and accurate product data by brand owners to consumers/shoppers, retailers, internet applications, and government using internet and mobile devices.

The vision of the TSD framework is that:

- Brand owners can share relevant product information easily, thus building trust with consumers.
- Internet Applications can ensure they are delivering authentic data.
- Consumers can feel confident that the digital product information they access is accurate, no matter how or where they shop.

In the envisioned solution, Internet Applications will interface with Data Aggregators to obtain trusted data regarding products identified with GS1 Identification Keys (namely, the Global Trade Item Number). Each Data Aggregator will draw upon a variety of sources for this data, including the GS1 Global Data Synchronisation Network (GDSN). Brand owners supply trusted data to the Data Aggregators through GDSN or other means.

4.1 Business Intention

The objective is to facilitate the emergence of a system of federated Data Aggregators, each capable of delivering trusted data about products to Internet Applications. Each Data Aggregator will draw upon a variety of sources for this data, including the GS1 Global Data Synchronisation Network (GDSN). Brand owners supply trusted data to the Data Aggregators through GDSN or other means.

It is anticipated that no one Data Aggregator will house data for all products. Therefore Data Aggregators will be federated by means of a Global Index, allowing each Data Aggregator to efficiently fetch missing product records from peer Data Aggregators.

4.2 Business Justification

The following business benefits are expected to be realised from the TSD framework:

- Consumers have accurate information (up-to-date, right product, consumer trust/confidence).
- Consumers are prevented from getting no information, inaccurate information or maliciously falsified information on a product.
- Consumers have relevant information, tailored to their preferences (profile) if needed.
- Brands have an easy way to provide content, either by themselves or in cooperation with certified third parties.
- Solution providers have an easy way to access content and show that it has been brand-authorised.

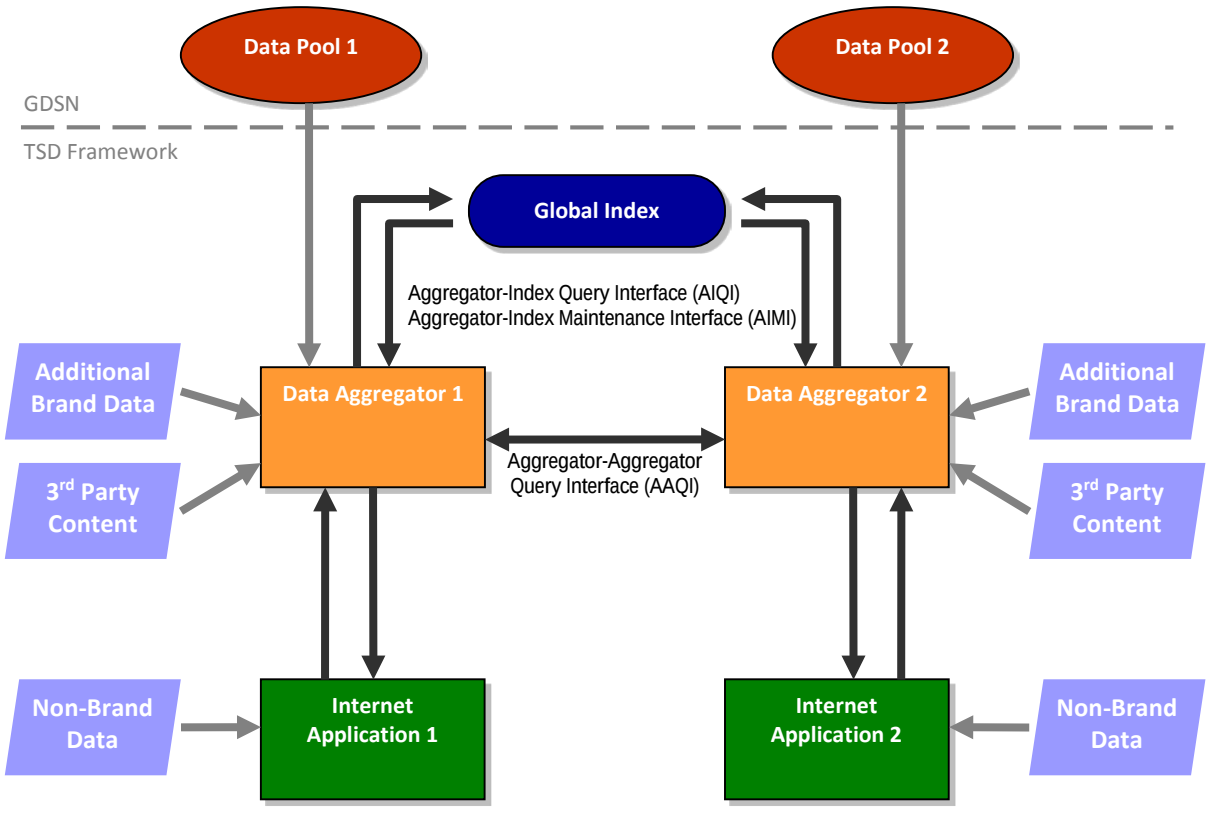
5 Architecture

This section defines the architecture of the Trusted Source of Data framework, providing the foundation for the normative definitions of data and interfaces specified in later sections.

5.1 Components and Interfaces

The Trusted Source of Data framework is based upon an architecture having the following relationships between networked computer systems:

Trusted Source of Data Framework Architecture



This architecture includes the following components:

- *Internet Application (one or more)* An application or service that is available to consumers via an Internet-based platform (desktop or mobile), and which queries a Data Aggregator to obtain product information
- *Data Aggregator (one or more)* A service that gathers or maintains data regarding products, and makes this data available to Internet Applications. Each Data Aggregator is not expected to have data for all products, but instead interacts with peer Data Aggregators as necessary to create a federated system.
- *Global Index (one)* A service that a Data Aggregator uses to identify peer Data Aggregators who have data about a given product, in the case where the first Data Aggregator does not itself have data about that product. Logically, there is one Global Index that serves all aggregators in the TSD framework; physically, this may be implemented by multiple services that are federated.
- *Data Pool (one or more)* A data pool in the GS1 Global Data Synchronisation Network. A Data Pool is the primary source of data that a Data Aggregator draws from.

The implementation of any given component is out of scope for this standard. This standard only defines the interfaces to which certain components must conform in order to participate in the TSD framework.

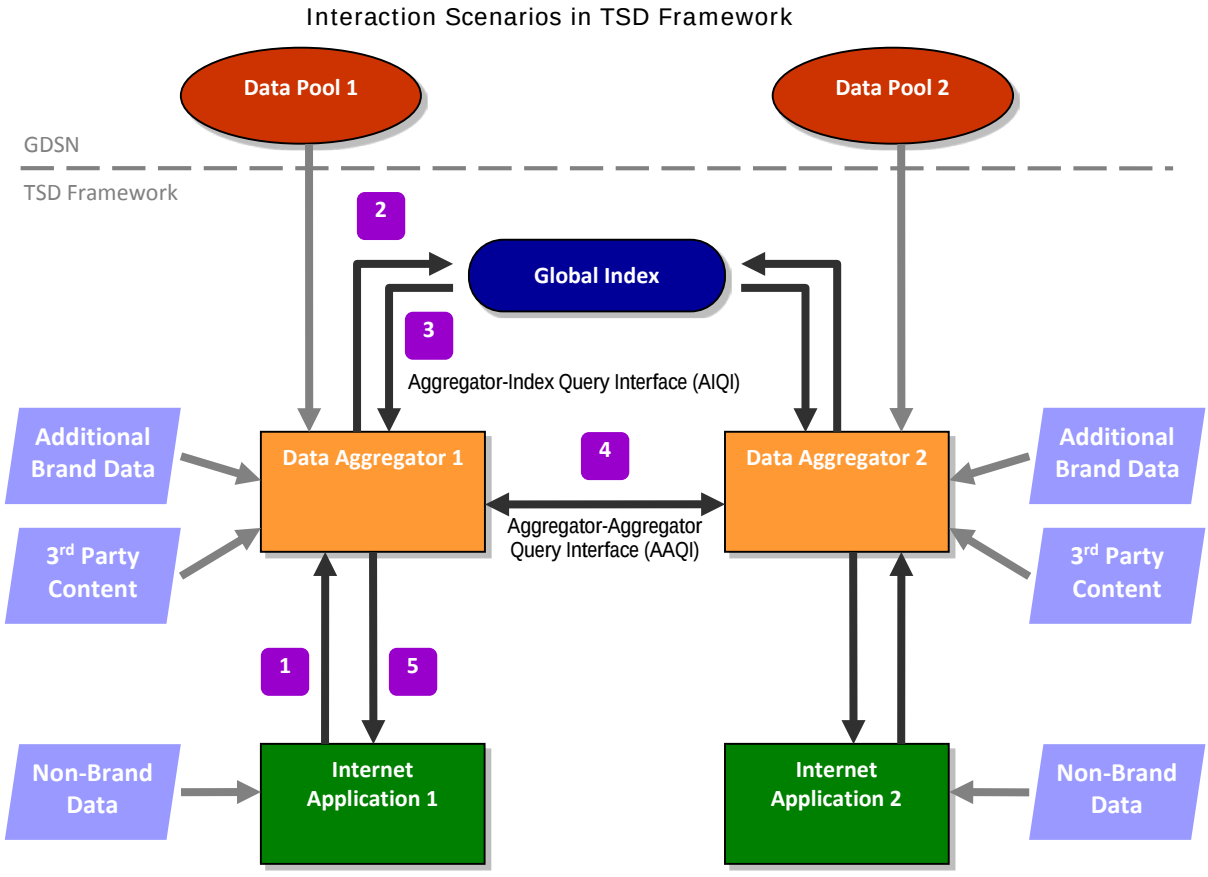
The scope of this standard is to define the following interfaces:

- *Aggregator-Aggregator Query Interface (AAQI)* The interface through which one Data Aggregator queries another for data about a specified product (or products).
- *Aggregator-Index Query Interface (AIQI)* The interface through which a Data Aggregator queries the Global Index to obtain references to other Data Aggregators who may have data about a specified product (or products).
- *Aggregator-Index Maintenance Interface (AIMI)* The interface through which a Data Aggregator interacts with the Global Index to add, change, correct, or delete index entries that reference the Data Aggregator.

Out of scope of this standard are the interactions between Data Aggregators and Internet Applications, and the interactions between Data Aggregators and GDSN Data Pools.

5.2 Interaction Scenarios

This section outlines two typical interactions in the TSD framework, to illustrate the collaboration between system components using the interfaces defined in this standard. Both scenarios make reference to the following figure:



5.2.1 Scenario 1: Data Delivered Directly from Local Data Aggregator

In this scenario, an Internet Application queries a Data Aggregator for product data for a specified product, and the Data Aggregator is able to satisfy the request using data it has locally.

Goal: To supply an Internet Application with product data for a specified product

Pre-condition: Internet Application and Data Aggregator have established a trust relationship so that they can mutually authenticate each other. Internet Application is authorised to query the Data Aggregator. Data Aggregator has the data for the specified product.

Post-condition: Internet Application has the requested data.

Main flow:

1. (0, label "1") Internet Application issues a query to the Data Aggregator, including a specification of what product information is desired (e.g., specifying the GTIN and Target Market in case of a query by GTIN).
2. Data Aggregator retrieves the requested data from its local sources of data.
3. (0, label "5") Data Aggregator responds to the Internet Application, providing the requested product data.

As this scenario only involves interaction between the Internet Application and the Data Aggregator, and the interactions between Internet Applications and Data Aggregators are out of scope for this standard (see Sections 1 and 5.1), this standard is not used at all in the carrying out of this scenario.

5.2.2 Scenario 2: Data Delivered from Peer Data Aggregator

In this scenario, an Internet Application queries a Data Aggregator for product data for a specified product, but the Data Aggregator is unable to satisfy the request using data it has locally, so it must interact with the Global Index and one or more peer Data Aggregators.

Goal: To supply an Internet Application with product data for a specified product

Pre-condition: Internet Application and Data Aggregator have established a trust relationship so that they can mutually authenticate each other. Internet Application is authorised to query the Data Aggregator. Data Aggregator has established trust relationships with the Global Index and peer Data Aggregators. Brand Owners have previously registered index entries with the Global Index for relevant products (a Brand Owner may authorise a Data Aggregator to perform index maintenance on its behalf)..

Post-condition: Internet Application has the requested data.

Main flow:

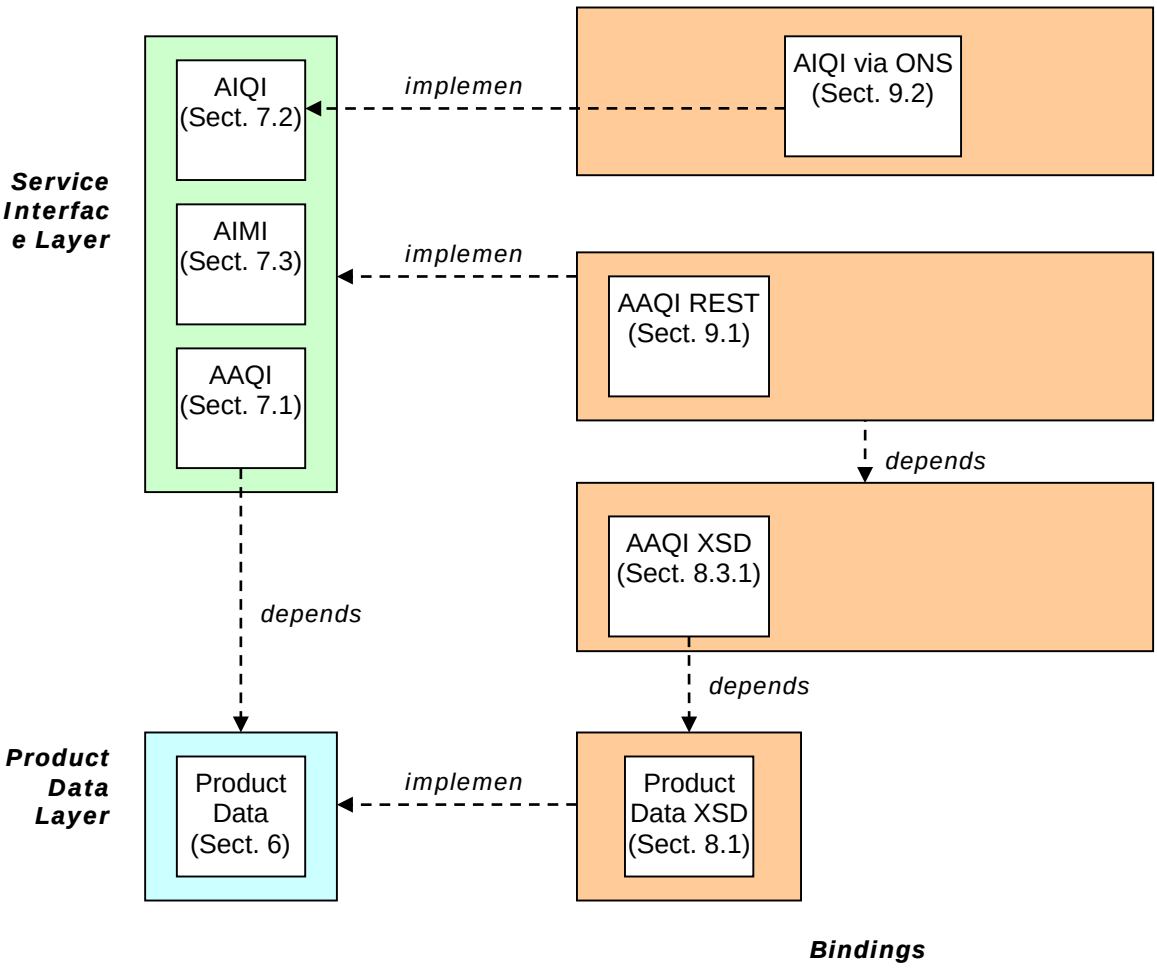
1. (0, label “1”) Internet Application issues a query to the Data Aggregator, including a specification of what product information is desired (e.g., specifying the GTIN and Target Market in case of a query by GTIN).
2. Data Aggregator finds that it does not have local data to satisfy the query.
3. (0, label “2”) Data Aggregator queries the Global Index to identify peer Data Aggregators that have relevant data.
4. (0, label “3”) Global Index responds with references to one or more Data Aggregators that have relevant data.
5. (0, label “4”) Data Aggregator queries peer Data Aggregator(s) to obtain relevant data.
6. (0, label “4”) Peer Data Aggregator(s) respond with relevant data.
7. (0, label “5”) Data Aggregator responds to the Internet Application, providing the requested product data assembled from data received from peer Data Aggregators.

Steps 3 and 4 of this flow (0, labels “2” and “3”) are governed by the Aggregator-Index Query Interface (AIQI), specified in Section 7.2. Steps 5 and 6 of this flow (0, label “4”) are governed by the Aggregator-Aggregator Query interface (AAQI), specified in Section 7.1.

5.3 Specification Layers

The TSD framework standard is organised into several layers, illustrated below.

TSD Framework Standard Layers



These layers are described below.

- *Product Data Layer* The Product Data Layer specifies the data exchanged through the Aggregator-Aggregator Query Interface: its abstract structure, and what it means.
- *Service Interface Layer* The Service Interface Layer defines the three service interfaces defined in Section 5.1: the Aggregator-Aggregator Query Interface (AAQI), the Aggregator-Index Query Interface (AIQI), and the Aggregator-Index Maintenance Interface (AIMI). This layer defines the abstract structure of these interfaces (what operations are contained within each interface), the abstract content of messages communicated in these interfaces to carry out the operations, and the semantics of each operation.
- *Bindings* Bindings specify concrete realisations of the Product Data Layer and the Service Interface Layer. In principle, many bindings may be defined for any given data definition or service interface. In this standard, an XML binding is given for the Product Data Layer; that is, an XML schema that concretely realises the abstract content of product data specified in the Product Data Layer. This standard also defines an XML binding for the messages in the AAQI, and this XML binding is used in turn to define a binding of the AAQI to a REST-style web service over HTTP. A binding for the AIQI is given in terms of ONS 2.0, which does not use XML. In this version of the standard, no bindings of the AIMI are specified. The abstract definition is provided as a guide for Data Aggregators and Global Index implementations that wish to provide a proprietary (non-

standard) implementation of the AIMI. Concrete bindings of AIMI may be specified in a future version of this standard.

Taken together, these layers are a complete specification of the data and interfaces within scope of this standard.

5.4 Versioning

The following mechanisms are provided to allow for evolution of this standard.

5.4.1 Versioning of Product Data Structure

The response to an AAQI query for product data is a `ProductData` structure as specified in Sections 6 and 8.1. Versioning of `ProductData` in this standard is designed to align with GS1 Global Data Synchronisation Network (GSDN) standards, and provides for the following types of changes:

- *Major Version* A new `ProductData` major version introduces a new XML schema which is not necessarily forward nor backward compatible with previous versions.
- *Minor Version* A new `ProductData` minor version introduces a new XML schema which is backwards compatible with earlier minor versions within the same major version, though not necessarily forward compatible. Backwards compatibility means that any data that is valid according to an older minor version is still valid in the new minor version and has the same meaning. Data that is valid in the new minor version may include new data elements that are not present in older minor versions.
- *Temporary Attributes* Between minor versions, it is possible to include additional attributes not defined in the XML schema for `ProductData`. Such attributes are included using a generic attribute-value-pair mechanism provided for in the XML schema. Attributes introduced in this way are resolved in the next version. When a temporary attribute is introduced, any Data Aggregator implementing the temporary attribute SHALL include it in `ProductData` it sends to other Data Aggregators.

A major or minor version is indicated by a version number. An AAQI query indicates the desired version number for data in the response. Normally, a Data Aggregator SHALL respond affirmatively to a query that specifies the current version and MAY respond affirmatively or reject a request specifying a different version. During a defined period of transition between an old version and a new version, a Data Aggregator SHALL respond affirmatively to a request specifying the old version and MAY respond affirmatively to a request specifying the new version or any other version.

During the defined period of transition, a querying Data Aggregator behaves as follows:

- If the querying Data Aggregator has not yet been upgraded to support the new version, it always issues a query specifying the old version. Because all Data Aggregators support queries for the old version during the transition period, they can all respond affirmatively.
- If the querying Data Aggregator has been upgraded to support the new version, it first issues a query specifying the new version. If the peer Data Aggregator responds negatively, the first Data Aggregator reissues the query specifying the old version. In this way, the upgraded Data Aggregator can interoperate with peers regardless of whether they have been upgraded yet.

5.4.2 Versioning of REST Interfaces

The REST interfaces for AAQI and AIMI use the following mechanisms to provide versioning for the request URL:

- *Compatible Change* New operations and new query parameters may be added to a REST interface by a new version of TSD. The base Service URL and service URL version number remains the same.
- *Incompatible Change* A REST interface may be changed incompatibly by updating the version string embedded in the request URL (see Section 9.1.1). During a period of transition, both the old and new URLs may be implemented simultaneously.

5.5 Conformance

The following types of implementation artefacts may conform to this standard.

- *Data Aggregator* A Data Aggregator implementation is in conformance to this standard if all of the following are true:
  - The implementation conforms to all normative statements in this standard that pertain to the Aggregator-Aggregator Query Interface (AAQI), both the querying side and the responding side, with respect to at least one binding of that interface.
  - The implementation conforms to all normative statements in this standard that pertain to the client side of the Aggregator-Index Query Interface (AIQI), with respect to at least one binding of that interface.
- *Global Index* A Global Index implementation is in conformance to this standard if the following is true:
  - The implementation conforms to all normative statements in this standard that pertain to the service side of the Aggregator-Index Query Interface (AIQI), with respect to at least one binding of that interface.

In addition to conformance to this standard, Data Aggregator implementations and Global Index implementations may be subject to service level agreements (SLAs) in order to participate in the GS1 TSD framework. Such SLAs are defined elsewhere.

Note that because the scope of this standard does not include any interfaces to which an Internet Application is a party, it is not valid to assert that an Internet Application is either in conformance or out of conformance to this standard. Individual Data Aggregator implementations may, however, choose to reuse the Product Data Layer of this standard in defining their own proprietary interfaces to Internet Applications, and in such cases a Data Aggregator may reference this standard in defining its own proprietary conformance criteria for its Internet Applications.

6 Product Data – Abstract Definition

This section defines product data that is communicated from one Data Aggregator to another in the Aggregator-Aggregator Query Interface (AAQI). This section defines the data content and meaning at an abstract level; see Section 8.1 for a concrete realisation of the data content as an XML schema.

6.1 Modular Structure

A given instance of product data provided by a Data Aggregator describes a single product (single GTIN) as sold in a single target market. The product data for a given GTIN + target market combination SHALL conform to the following:

- The product data SHALL include Basic Product Information as specified in [GS1PDM] section 4.1.
- The product data MAY include other modules. If it does, these modules SHALL be as specified in [GS1PDM] Section 4 and further.

The UML diagram in Section 6.3 expresses the high level structure of product data.

6.2 Common Data Types

This section defines data types used by all product data modules and interfaces.

6.2.1 GTIN

The `GTIN` type represents a Global Trade Item Number (GTIN). The `GTIN` type is a 14-character string which may contain a GTIN-8, GTIN-12, GTIN-13, or GTIN-14 as defined in the GS1 General Specifications [GS1GS]. When the `GTIN` type holds a GTIN-8, GTIN-12, or GTIN-13, the GTIN value is padded on the left with zeros to make 14 characters total, as illustrated in [GS1GS, Section 3.3.2].

6.2.2 CountryCode

The `CountryCode` type is a 3-character numeric string that identifies a country, using the 3-digit country codes defined by ISO 3166-1 numeric [ISO3166].

The `CountryCode` type is used in the TSD standard to denote a target market for a GTIN.

6.2.3 LanguageCode

The `LanguageCode` type is a 2-character string that identifies a human language, using the 2-character language codes defined by ISO 639-1 [ISO639]. Note that both characters of an ISO 639-1 code are lowercase.

6.2.4 ServiceReference

A `ServiceReference` is used in the response to a Global Index query to refer to a specific Data Aggregator service. A `ServiceReference` is an absolute HTTP URL that is used as the base URL in an AAQI query (see Section 9.1.1).

6.2.5 GLN

The `GLN` type represents a Global Location Number (GLN). The `GLN` type is a 13-character string.

6.2.6 Description70, Description80, Description200, Description2500

The types `Description70`, `Description80`, `Description200`, and `Description2500` are a pair consisting of a text string and a `LanguageCode` (Section 6.2.3). The maximum length of the text string is 70, 80, 200, or 2500 characters, respectively.

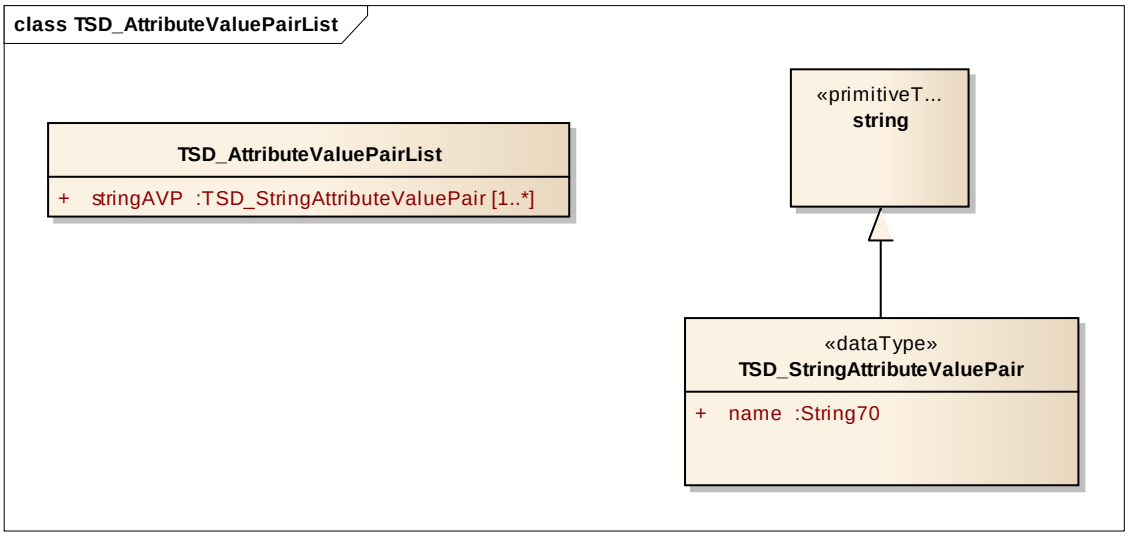
These types are used within product data where a human-readable string may be expressed in more than one language. In each such place more than one element of the chosen type is permitted, to allow for a description to be provided in multiple languages.

6.2.7 Measurement

The `Measurement` type is a pair consisting of a value of type Float and a unit of measure of type String. The value of the unit of measure SHALL be one of the units of measure specified in UN/ECE Recommendation 20 [UNECE20], specifically the string defined in the “Common Code” column of [UNECE20] for the selected unit.

6.2.8 TSD_AttributeValuePairList

Attribute-value pair lists are used to provide a means to introduce temporary attributes (see Section 5.4.1) into product data between minor versions. The following UML diagram specifies the structure of an attribute-value pair list:



An attribute value pair list consists of one or more attribute-value pairs, each of which contains a name and a value.

To avoid clutter in UML diagrams in this standard, references to attribute value pair lists are shown as UML attributes rather than associations; that is, a class containing an attribute value pair list includes a UML attribute of type `TSD_AttributeValuePairList` rather than showing an arrow to the `TSD_AttributeValuePairList` class. The UML stereotype `<<association>>` is used to mark this deviation from normal GS1 UML design rules. It has no effect on the XML schemas.

6.2.9 TSD_Formatted Description1000, 2500, 5000

The types `TSD_FormattedDescription1000`, `TSD_FormattedDescription2500`, `TSD_FormattedDescription5000` consist of a value of type string, a `LanguageCode` (Section 6.2.3) and an optional `formattingPattern`. The maximum length of the string is 1000, 2500, or 5000 characters, respectively.

In case `formattingPattern` is present, it SHALL contain one or more 'p' and 'em' formatting elements.

Each formatting element SHALL have an opening and closing position with the following structure:

`(startingPosition,elementName)(closingPosition,/elementName)`

The `startingPosition` specifies the first character of the text that needs to be formatted, the `closingPosition` specifies the last character of the text that needs to be formatted.

The 'p' and 'em' elements SHALL be interpreted and used in accordance with the W3C HTML 4.01 Specification, specifically:

A `formattingPattern` SHALL contain one or more 'p' formatting elements, covering the full text string. When multiple 'p'elements are present they SHALL not have overlapping ranges, and they SHALL be specified in the sequence of the opening position.

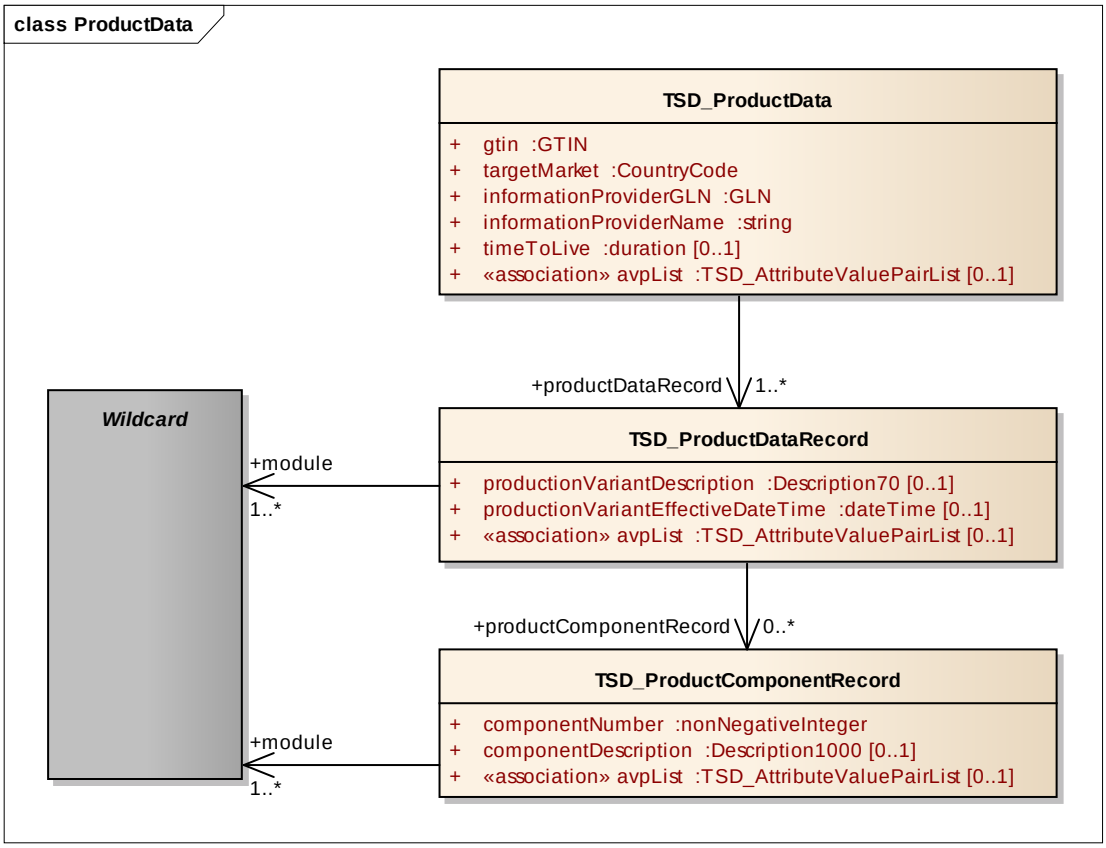
Each 'p' element MAY contain one or more 'em' elements. These SHALL be included within the opening and closing position of the 'p' element. When multiple 'em' elements are present they SHALL not have overlapping ranges, and they SHALL be specified in the sequence of the opening position.

Example:

```
<ingredientStatement formattingPattern="(1,p)(9,em)(12,/em)(22,/p)">tomato,
nuts, cucumber</ingredientStatement>
```

6.3 Product Data

The following UML diagram expresses the data content of Product Data. This is the structure that one Data Aggregator sends to another in response to a query issued via the Aggregator-Aggregator Query Interface (AAQI). See Section 7.1.



The data content of a `TSD_ProductData` structure SHALL be as follows:

Data Element	Type	Cardinality	Description
`gtin`	`GTIN` (Section 6.2.1)	1	The GTIN of the product described by this `TSD_ProductData` instance.
`targetMarket`	`CountryCode` (Section 6.2.2)	1	Target market to which this `TSD_ProductData` instance applies, expressed as a `CountryCode` (Section 6.2.2).
`informationProviderGLN`	`GLN` (Section 6.2.5)	1	Party Global Location Number (GLN) of the party providing this data

Data Element	Type	Cardinality	Description
informationProviderName	String	1	Human-readable name of the party providing this data.
timeToLive	Duration	0..1	If present, indicates that the data is only valid within the specified duration after receipt by the requesting Data Aggregator.
productDataRecord	TSD_ProductDataRecord (below)	1..*	Product data records describing the product identified by the specified GTIN for the specified target market. If more than one product data record is included, each describes a different variant of the product.
avpList	TSD_AttributeValuePairList (Section 6.2.8)	0..1	Temporary attributes introduced between minor versions; see Section 5.4.1.

The data content of a TSD_ProductDataRecord SHALL be as follows:

Data Element	Type	Cardinality	Description
productionVariantDescription	Description70 (Section 6.2.6)	0..1	Free text assigned by the manufacturer to describe the production variant. Examples are: package series X, package series Y.
productionVariantEffectiveDateTime	DateTime	0..1	The start date of a production variant.
Module	Product data module type; one or more of the types specified in Section 6.4	1..*	Product data modules. The first module SHALL be the basic product information module. Other modules are optional. This list SHALL contain at most one module of each module type.

Data Element	Type	Cardinality	Description
avpList	TSD_AttributeValue PairList (Section 6.2.8)	0..1	Temporary attributes introduced between minor versions; see Section 5.4.1.
productComponentRecord	TSD_ProductComponent Record (below)	0..*	Product component records describing components of the product. If more than one product component record is included, each describes a different component of the product.

The data content of a TSD_ProductComponentRecord SHALL be as follows:

Data Element	Type	Cardinality	Description
componentNumber	nonNegativeInteger	1..1	Sequence number that uniquely identifies the component record.
componentDescription	Description1000 (Section 6.2.6)	0..*	Free text assigned by the manufacturer to describe the component. Example flavor X, Y, Z.
module	Product data module type; one or more of the types specified in Section 6.4	0..*	Product data modules. This list SHALL contain at most one module of each module type.
avpList	TSD_AttributeValue PairList (Section 6.2.8)	0..1	Temporary attributes introduced between minor versions; see Section 5.4.1.

6.4 Product Data Module Types

The product data module types that may occur in the modules list of a ProductData structure (Section 6.3) are defined in a separate document, see [TSDPDM]

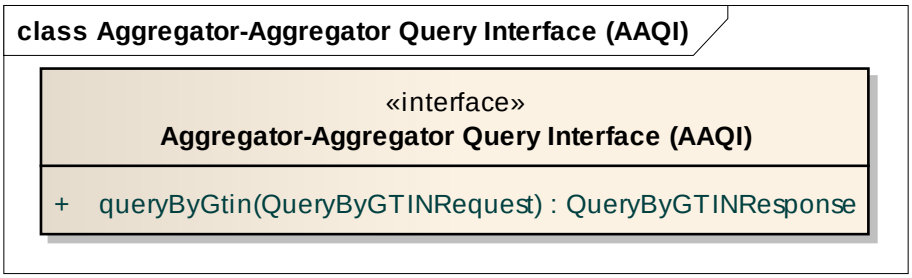
7 Interfaces – Abstract Definition

This section defines the three service interfaces that constitute this standard: the Aggregator-Aggregator Query Interface (AAQI), the Aggregator-Index Query Interface (AIQI), and the Aggregator-Index Maintenance Interface (AIMI). This section defines the operations within each interface and their semantics at an abstract level. See Section 8.3 for concrete realisations as an XML schema of the request and response messages implied by the interface definitions, Section 9.1for a concrete realisation as web services of these interfaces based on the XML schemas in Section 8.3, and Section 9.2 for a (non-XML) implementation of the AIQI based on ONS 2.0.

7.1 Aggregator-Aggregator Query Interface (AAQI) – Abstract Definition

This section specifies the Aggregator-Aggregator Query Interface (AAQI). A Data Aggregator SHALL implement both the requesting and responding side of this interface.

The following UML diagram expresses the operations in the AAQI.



A Data Aggregator SHALL implement the following interface operations:

Operation	Section	Description
queryByGtin	7.1.1	Retrieve product data for a specified GTIN and target market, subject to specified language preferences.

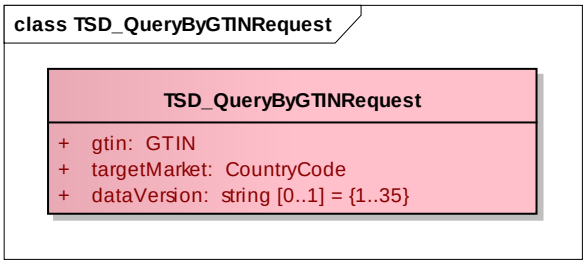
A Data Aggregator SHALL implement the following interface exceptions:

Exception	Operations	Description
NoDataException	queryByGtin	No data is available for the GTIN + target market combination specified in the request
InvalidGtinException	queryByGtin	The GTIN specified in the request is not properly formatted, or has an incorrect check digit.
InvalidTargetMarket Exception	queryByGtin	The target market specified in the request is not properly formatted
UnsupportedVersion Exception	queryByGtin	The ProductData version specified in the request is not supported (see Section 5.4.1)
InvalidRequest Exception	queryByGtin	The request was not formatted correctly, contained an unknown parameter, or specified an unknown operation.
SecurityException	queryByGtin	The operation was not permitted due to an access control violation or other security concern. This includes the case where the service wishes to deny authorisation to execute a particular operation based on the authenticated client identity.
Implementation Exception	queryByGtin	A generic exception thrown by the implementation for reasons that are implementation-specific, such as a software error.

If more than one exception applies to a given request, the Data Aggregator SHALL respond with the applicable exception that occurs closer to the bottom of the above list; however the Data Aggregator MAY choose to respond with `InvalidRequestException` in the case that `InvalidRequestException` and `SecurityException` both apply.

7.1.1 AAQI queryByGtin Operation

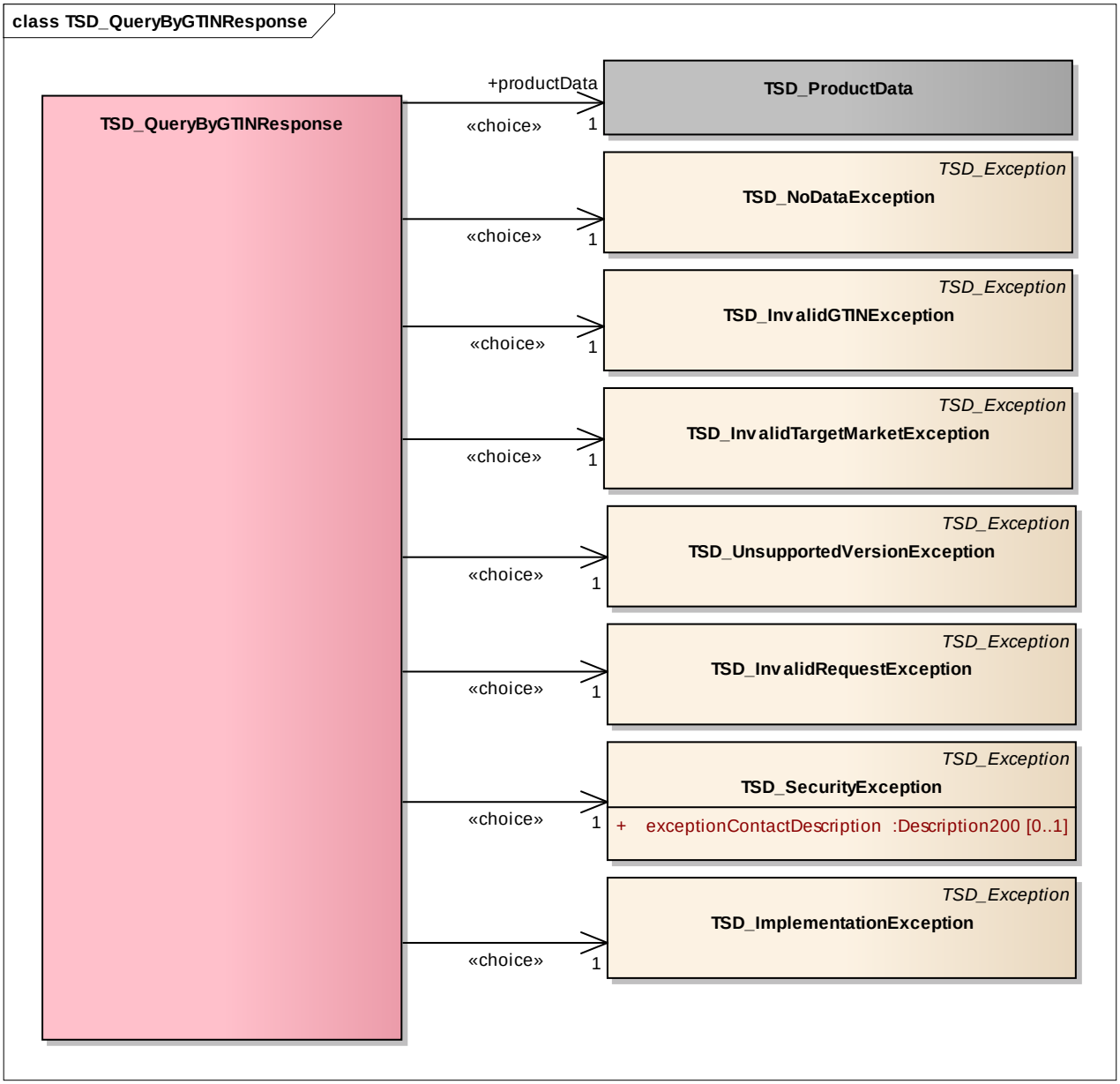
The following UML diagram expresses the data content of the request for the `queryByGtin` operation of the AAQI.



The data content of the `TSD_QueryByGtinRequest` SHALL be as follows:

Data Element	Type	Cardinality	Description
`gtin`	`GTIN` (Section 6.2.1)	1	The GTIN of the product being requested.
`targetMarket`	`CountryCode` (Section 6.2.2)	1	Target market for the data being requested, expressed as a `CountryCode` (Section 6.2.2).
`dataVersion`	`String`	1	The version number of the desired version of `ProductData`

The following UML diagram expresses the data content of the response for the QueryByGtin operation of the AAQI.



The data content of the TSD_QueryByGtinResponse SHALL be either a TSD_ProductData structure (Section 6) or one of the exceptions specified above.

The data content of each exception *except* TSD_SecurityException SHALL be as follows:

Data Element	Type	Cardinality	Description
exceptionReason (provided in base class TSD_Exception)	Description200 (Section 6.2.6)	1	A human readable string providing additional information about the exception. This is not intended for presentation to end users. The content is at the discretion of the responding Data Aggregator but SHOULD always be in English

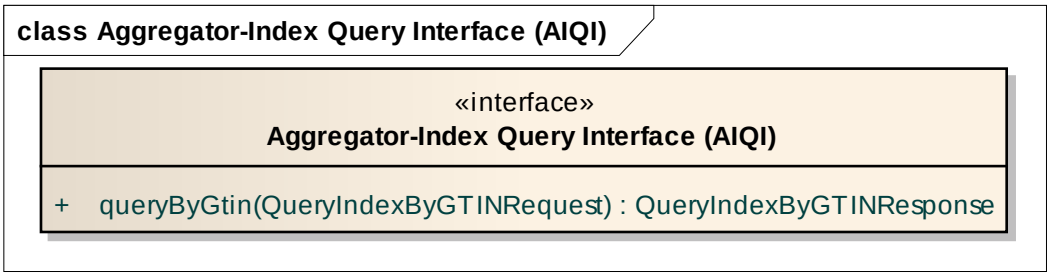
The data content of TSD_SecurityException SHALL be as follows:

Data Element	Type	Cardinality	Description
exceptionReason (provided in base class TSD_Exception)	Description200 (Section 6.2.6)	1	A human readable string providing additional information about the exception. This is not intended for presentation to end users. The content is at the discretion of the responding Data Aggregator but SHOULD always be in English
exceptionContactDescription	Description200 (Section 6.2.6)	0..1	A human readable string providing contact information for establishing a trust relationship with a Data Aggregator not previously encountered.

7.2 Aggregator-Index Query Interface (AIQI) – Abstract Definition

This section specifies the Aggregator-Index Query Interface (AIQI). A Data Aggregator SHALL implement the client side of this interface. A Global Index SHALL implement the server side of this interface.

The following UML diagram expresses the operations in the AIQI.



A Data Aggregator (client side) or Global Index (server side) SHALL implement the following interface operations:

Operation	Section	Description
`queryByGtin`	7.2.1	Retrieve index data for a specified GTIN and target market.

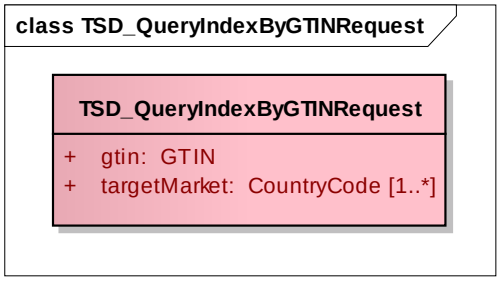
A Data Aggregator (client side) or Global Index (server side) SHALL implement the following interface exceptions:

Exception	Operations	Description
`InvalidGtinException`	`queryByGtin`	The GTIN specified in the request is not properly formatted, or has an incorrect check digit.
`InvalidTargetMarket Exception`	`queryByGtin`	The target market specified in the request is not properly formatted
`InvalidRequest Exception`	`queryByGtin`	The request was not formatted correctly, contained an unknown parameter, or specified an unknown operation.
`SecurityException`	`queryByGtin`	The operation was not permitted due to an access control violation or other security concern. This includes the case where the service wishes to deny authorisation to execute a particular operation based on the authenticated client identity.
`Implementation Exception`	`queryByGtin`	A generic exception thrown by the implementation for reasons that are implementation-specific, such as a software error.

If more than one exception applies to a given request, the Global Index SHALL respond with the applicable exception that occurs closer to the bottom of the above list; ; however the Global Index MAY choose to respond with `InvalidRequestException` in the case that `InvalidRequestException` and `SecurityException` both apply..

7.2.1 AIQI queryByGtin

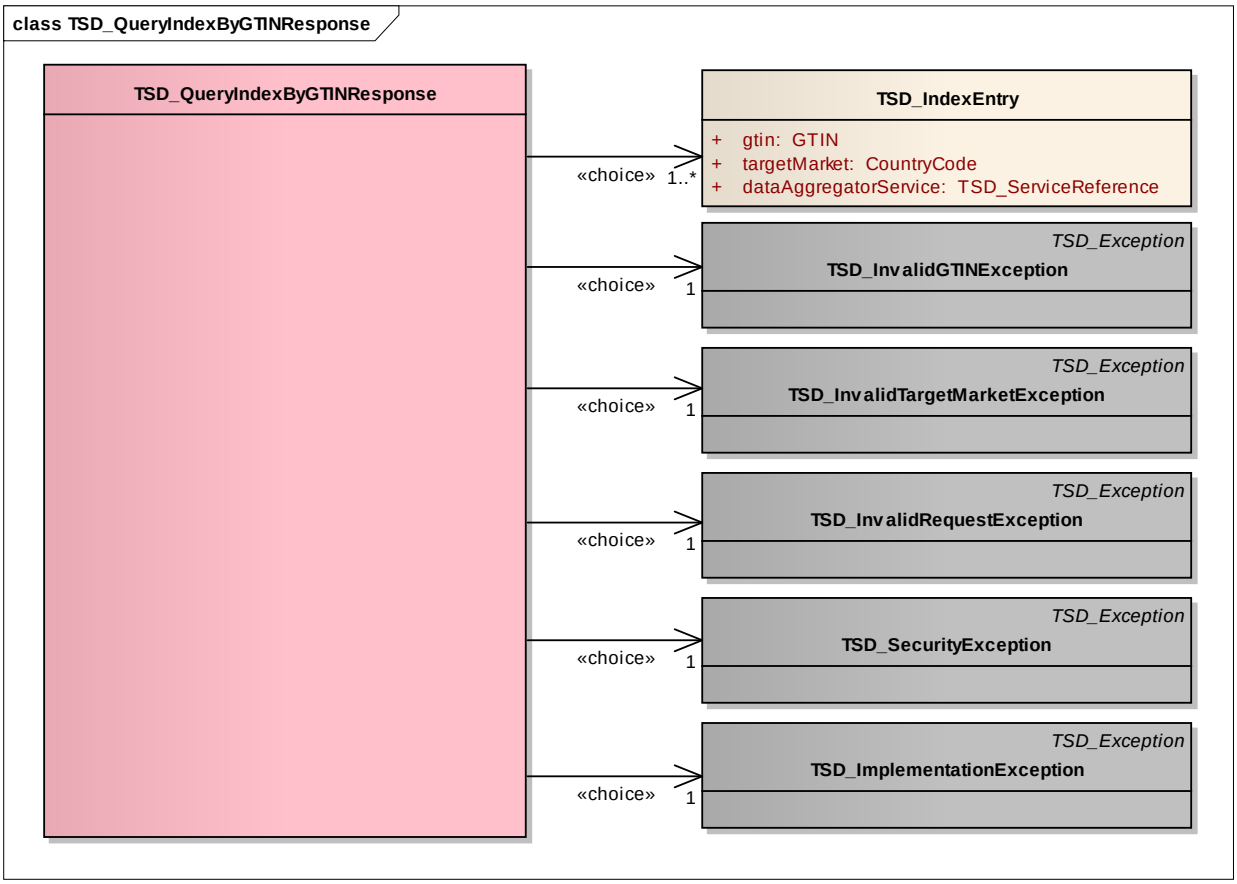
The following UML diagram expresses the data content of the request for the `queryByGtin` operation of the AIQI.



The data content of the `TSD_QueryIndexByGtinRequest` SHALL be as follows:

Data Element	Type	Cardinality	Description
`gtin`	`GTIN` (Section 6.2.1)	1	The GTIN of the product being requested.
`targetMarket`	`CountryCode` (Section 6.2.2)	1..*	Target markets for the data being requested, each expressed as a `CountryCode` (Section 6.2.2).

The following UML diagram expresses the data content of the response for the `queryByGtin` operation of the AIQI.



The data content of the `TSD_QueryIndexByGtinResponse` SHALL be either a list of zero or more `IndexEntry` structures that match the GTIN and target market(s) specified in the request, or be one of the exceptions defined above.

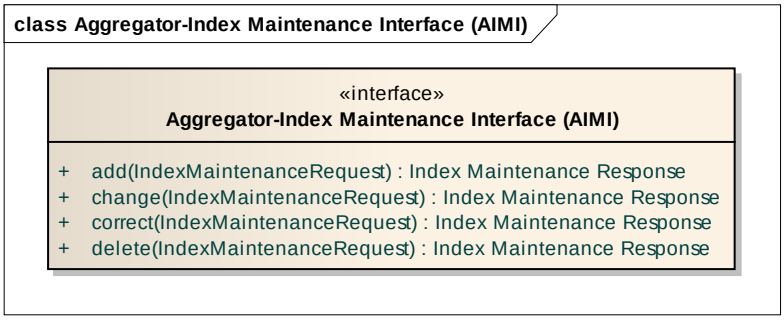
The data content of an IndexEntry SHALL be as follows:

Data Element	Type	Cardinality	Description
gtin	GTIN (Section 6.2.1)	1	The GTIN of the product referenced by this index entry.
targetMarket	CountryCode (Section 6.2.2)	1	Target market for the data referenced by this index entry, expressed as a CountryCode (Section 6.2.2).
dataAggregatorService	ServiceReference (Section 6.2.4)	1	Reference to a Data Aggregator service that has data for the GTIN, target market combination specified in this index entry.

7.3 Aggregator-Index Maintenance Interface (AIMI) – Abstract Definition

This section specifies the Aggregator-Index Maintenance Interface (AIMI). A Data Aggregator MAY implement the client side of this interface. A Global Index MAY implement the server side of this interface. In this version of the standard, no binding of this abstract interface to a concrete service is specified. The abstract definition is provided as a guide for Data Aggregators and Global Index implementations that wish to provide a proprietary (non-standard) implementation of this interface. Concrete bindings of AIMI interface may be specified in a future version of this standard.

The following UML diagram expresses the operations in the AIMI.



A Data Aggregator (client side) or Global Index (server side) SHALL implement the following interface operations:

Operation	Description
add	Add a new index entry for a specified GTIN and target market.
Change	Change an existing index entry for a specified GTIN and target market.
Correct	Correct an existing index entry for a specified GTIN and target market. The effect of this operation is identical to the change operation, but expresses a different intent on the part of the client.
Delete	Delete an index entry for a specified GTIN and target market.

At most one index entry exists for each unique combination of gtin, target market, and data aggregator service. Note that the AIMI is designed to support maintenance of index entries that may contain additional searchable attributes besides gtin, target market, and data aggregator. The change and correct operations provide a means to alter the additional searchable attributes for

an entry that already exists for a given gtin, target market, and data aggregator. As specified below, however, there are no such additional searchable attributes and the AIQI only supports query by gtin and target market. Without additional searchable attributes, the `change` and `correct` operations have no effect.

A Data Aggregator (client side) or Global Index (server side) SHALL implement the following interface exceptions:

Exception	Operations	Description
`DuplicateEntryException`	`add`	There already exists an entry for the specified GTIN, target market, and data aggregator service.

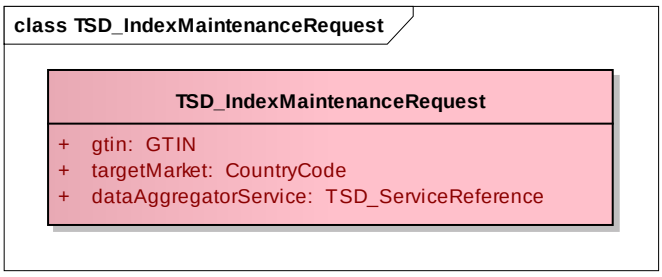
`NoSuchEntryException`	`change`
	`correct`
	`delete`
		No entry exists for the specified GTIN, target market, and data aggregator service.

`InvalidGtinException`	(all)	The GTIN specified in the request is not properly formatted, or has an incorrect check digit.
`InvalidTargetMarket Exception`	(all)	The target market specified in the request is not properly formatted.
`InvalidServiceURL Exception`	(all)	The data aggregator service URL specified in the request is not an absolute URL as specified in [RFC3986].
`InvalidRequestException`	(all)	The request was not formatted correctly, contained an unknown parameter, or specified an unknown operation.
`SecurityException`	(all)	The operation was not permitted due to an access control violation or other security concern. This includes the case where the service wishes to deny authorisation to execute a particular operation based on the authenticated client identity.
`ImplementationException`	(all)	A generic exception thrown by the implementation for reasons that are implementation-specific, such as a software error.

If more than one exception applies to a given request, the Global Index SHALL respond with the applicable exception that occurs closer to the bottom of the above list; however the Global Index MAY choose to respond with `InvalidRequestException` in the case that `InvalidRequestException` and `SecurityException` both apply..

7.3.1 AIMI Index Maintenance Request

The following UML diagram expresses the data content of the request for all AIMI operations.



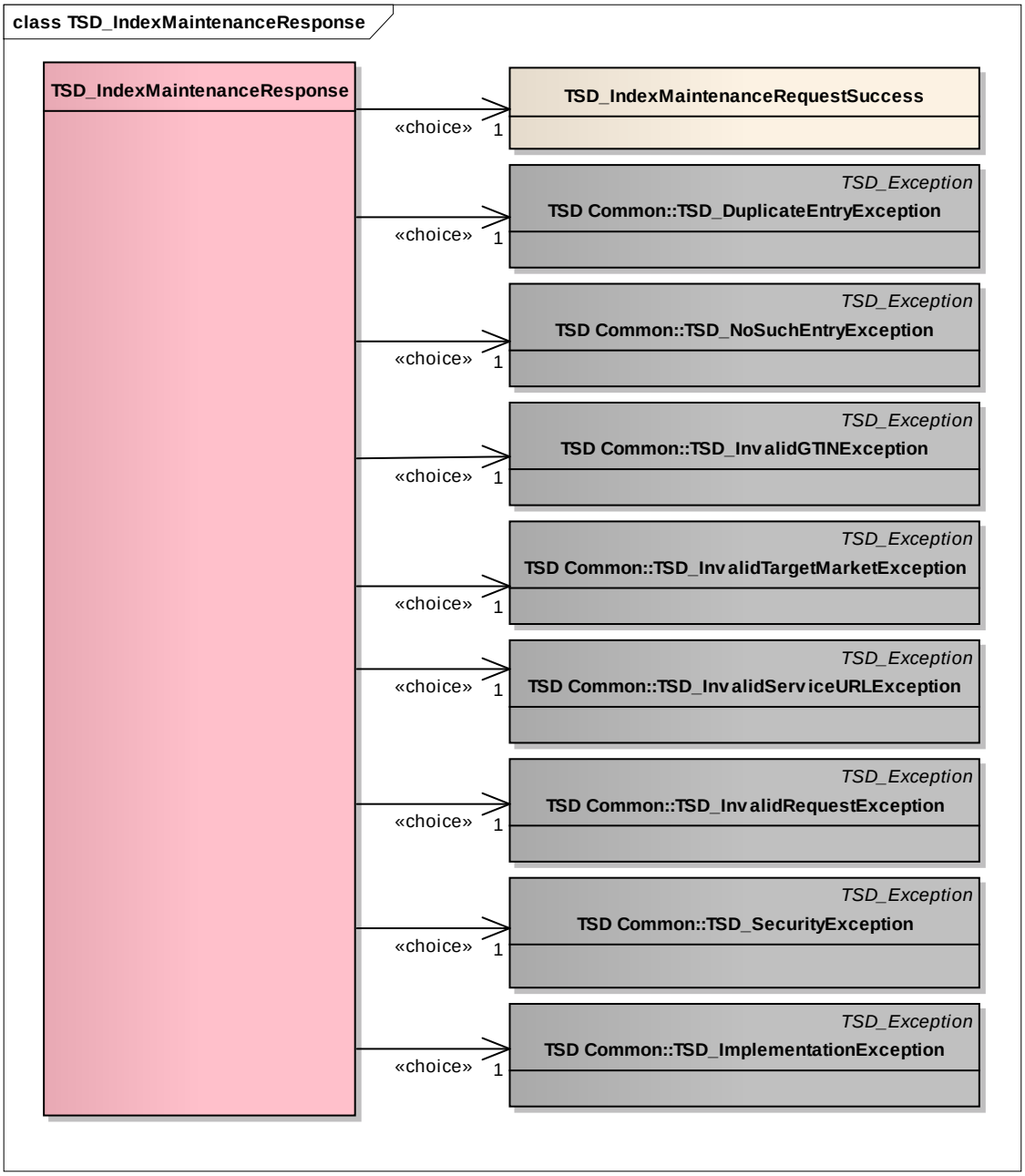
The data content of a TSD_IndexMaintenanceRequest SHALL be as follows:

Data Element	Type	Cardinality	Description
gtin	GTIN (Section 6.2.1)	1	The GTIN of the product referenced by this index entry.
targetMarket	CountryCode (Section 6.2.2)	1	Target market for the data referenced by this index entry, expressed as a CountryCode (Section 6.2.2).
dataAggregatorService	ServiceReference (Section 6.2.4)	1	Reference to a Data Aggregator service that has data for the GTIN, target market combination specified in this index entry.

7.3.2

7.3.3 AIMI Index Maintenance Response

The data content of the `IndexMaintenanceResponse` SHALL be either an `IndexMaintenanceSuccess` structure or one of the exceptions defined above.



The data content of each exception *except* TSD_SecurityException SHALL be as follows:

Data Element	Type	Cardinality	Description
exceptionReason (provided in base class TSD_Exception)	Description200 (Section 6.2.6)	1	A human readable string providing additional information about the exception. This is not intended for presentation to end users. The content is at the discretion of the responding Data Aggregator but SHOULD always be in English

The data content of TSD_SecurityException SHALL be as follows:

Data Element	Type	Cardinality	Description
exceptionReason (provided in base class TSD_Exception)	Description200 (Section 6.2.6)	1	A human readable string providing additional information about the exception. This is not intended for presentation to end users. The content is at the discretion of the responding Data Aggregator but SHOULD always be in English
exceptionContactDescription	Description200 (Section 6.2.6)	0..1	A human readable string providing contact information for establishing a trust relationship with a Data Aggregator not previously encountered.

8 XML Schemas

This section specifies XML schemas that implement the data definitions specified in previous sections. These schemas are referenced normatively by the definitions of XML-based service interface bindings specified in Section 9.1.

The XML data returned in the REST binding of AAQI as specified in Section 7.1 SHALL conform to the schemas in this section.

8.1 Common Data Types XML Schema

This section specifies two XML schemas that implements the common data types specified in Section 6.1.

8.1.1 Common Data Types XML Schema – Shared

This section specifies an XML schema that implements common data types specified in Section 6.1 that are also used in other GS1 standards. Only those common data types used by this TSD standard are included in the schema below; the published XSD file accompanying this standard may include additional type definitions not used by TSD.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:shared_common="urn:gs1:shared:shared_common:xsd:3"
targetNamespace="urn:gs1:shared:shared_common:xsd:3" elementFormDefault="unqualified"
attributeFormDefault="unqualified" version="3.1">
  <xsd:annotation>
    <xsd:documentation><![CDATA[---------------------------
© Copyright GS1, 2015

GS1 is providing this XML Schema Definition file and resultant XML file as a service to
interested industries.
This XML Schema Definition file and resultant XML file were developed through a consensus
process of interested parties.

Although efforts have been made to ensure that the XML Schema Definition file and resultant XML
file are correct, reliable, and technically
accurate,  GS1 makes NO WARRANTY, EXPRESS OR IMPLIED, THAT THIS XML Schema Definition file and
resultant XML file ARE
CORRECT, WILL NOT REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE, OR
WILL BE SUITABLE FOR
ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR OTHERWISE.  Use of the XML Schema Definition
file and resultant XML
file are with the understanding that GS1 has no liability for any claim to the contrary, or for
any damage or loss of any kind or nature.

Version Information:
Version Number: 3.1
Date of creation: April 2013

The schema and subsequent updates will be provided on the GS1 websites.
---------------------------
]]></xsd:documentation>
  </xsd:annotation>
  <xsd:complexType name="CommunicationChannelCodeType">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:GS1CodeType"/>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="CommunicationChannelType">
    <xsd:sequence>
      <xsd:element name="communicationChannelCode"
type="shared_common:CommunicationChannelCodeType"/>
      <xsd:element name="communicationValue">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="200"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="communicationChannelName" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="200"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CountryCodeType">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:GS1CodeType"/>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="CountrySubdivisionCodeType">
    <xsd:simpleContent>
```

```
    <xsd:extension base="shared_common:GS1CodeType"/>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="Description1000Type">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String1000Type">
      <xsd:attribute name="languageCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="Description200Type">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String200Type">
      <xsd:attribute name="languageCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="Description2500Type">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String2500Type">
      <xsd:attribute name="languageCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="Description5000Type">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String5000Type">
      <xsd:attribute name="languageCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
```

```
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="Description500Type">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String500Type">
      <xsd:attribute name="languageCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="Description70Type">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String70Type">
      <xsd:attribute name="languageCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="Description80Type">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String80Type">
      <xsd:attribute name="languageCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
```

```
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="ExtensionType">
  <xsd:sequence>
    <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GS1CodeType">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:String80Type">
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="LanguageCodeType">
  <xsd:simpleContent>
    <xsd:extension base="shared_common:GS1CodeType"/>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="MeasurementType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:decimal">
      <xsd:attribute name="measurementUnitCode" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="codeListVersion">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="35"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="GLNType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{13}"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="GTINType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{14}"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="String1000Type">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="1000"/>
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="String200Type">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="200"/>
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
  <xsd:simpleType name="String2500Type">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="2500"/>
      <xsd:minLength value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="String5000Type">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="5000"/>
      <xsd:minLength value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="String500Type">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="500"/>
      <xsd:minLength value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="String70Type">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="70"/>
      <xsd:minLength value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="String80Type">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="80"/>
      <xsd:minLength value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

8.1.2 Common Data Types XML Schema – TSD-specific

This section specifies an XML schema that implements common data types specified in Section 6.1 that are only used in this TSD standard (and not any other GS1 standard).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:shared_common="urn:gs1:shared:shared_common:xsd:3"
xmlns:tsd_common="urn:gs1:tsd:tsd_common:xsd:1" targetNamespace="urn:gs1:tsd:tsd_common:xsd:1"
elementFormDefault="unqualified" attributeFormDefault="unqualified" version="1.1">
  <xsd:annotation>
    <xsd:documentation><![CDATA[---------------------------
© Copyright GS1, 2015

GS1 is providing this XML Schema Definition file and resultant XML file as a service to
interested industries.
This XML Schema Definition file and resultant XML file were developed through a consensus
process of interested parties.

Although efforts have been made to ensure that the XML Schema Definition file and resultant XML
file are correct, reliable, and technically
accurate,  GS1 makes NO WARRANTY, EXPRESS OR IMPLIED, THAT THIS XML Schema Definition file and
resultant XML file ARE
CORRECT, WILL NOT REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE, OR
WILL BE SUITABLE FOR
ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR OTHERWISE.  Use of the XML Schema Definition
file and resultant XML
file are with the understanding that GS1 has no liability for any claim to the contrary, or for
any damage or loss of any kind or nature.

Version Information:
Version Number: 1.1
Date of creation: June 2013

The schema and subsequent updates will be provided on the GS1 websites.
---------------------------
]]></xsd:documentation>
  </xsd:annotation>
  <xsd:import namespace="urn:gs1:shared:shared_common:xsd:3"
schemaLocation="../shared/SharedCommon.xsd"/>
  <xsd:complexType name="TSD_AttributeValuePairListType">
    <xsd:sequence>
```

```
      <xsd:element name="stringAVP" type="tsd_common:TSD_StringAttributeValuePairType"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TSD_ExceptionType">
    <xsd:sequence>
      <xsd:element name="exceptionReason" type="shared_common:Description200Type"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TSD_FormattedDescription1000Type">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:String1000Type">
        <xsd:attribute name="languageCode" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="80"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="formattingPattern">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="500"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_FormattedDescription2500Type">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:String2500Type">
        <xsd:attribute name="languageCode" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="80"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="formattingPattern">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="500"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_FormattedDescription5000Type">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:String5000Type">
        <xsd:attribute name="languageCode" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="80"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="formattingPattern">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="500"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

```
  <xsd:complexType name="TSD_ImageLinkType">
    <xsd:sequence>
      <xsd:element name="url">
        <xsd:simpleType>
          <xsd:restriction base="xsd:anyURI">
            <xsd:maxLength value="2500"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="imageTypeCode" type="tsd_common:TSD_ImageTypeCodeType"/>
      <xsd:element name="languageCode" type="shared_common:LanguageCodeType" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="imagePixelHeight" type="xsd:nonNegativeInteger" minOccurs="0"/>
      <xsd:element name="imagePixelWidth" type="xsd:nonNegativeInteger" minOccurs="0"/>
      <xsd:element name="fileSize" type="shared_common:MeasurementType" minOccurs="0"/>
      <xsd:element name="avpList" type="tsd_common:TSD_AttributeValuePairListType"
minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TSD_ImageTypeCodeType">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:GS1CodeType"/>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_MeasurementPrecisionCodeType">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:GS1CodeType"/>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_ProductInformationLinkType">
    <xsd:sequence>
      <xsd:element name="url">
        <xsd:simpleType>
          <xsd:restriction base="xsd:anyURI">
            <xsd:maxLength value="2500"/>
            <xsd:minLength value="1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="productInformationTypeCode"
type="tsd_common:TSD_ProductInformationTypeCodeType"/>
      <xsd:element name="languageCode" type="shared_common:LanguageCodeType" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="avpList" type="tsd_common:TSD_AttributeValuePairListType"
minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TSD_ProductInformationTypeCodeType">
    <xsd:simpleContent>
      <xsd:extension base="shared_common:GS1CodeType"/>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_StringAttributeValuePairType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="70"/>
              <xsd:minLength value="1"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

8.2 Product Data XML Schema

This section specifies an XML schema for the top level of Product Data as specified in Section 6.3. The module element, which may be repeated any number of times, is specified as an XSD wildcard, but in actual instance data each occurrence SHALL be populated by an instance of one of the module schemas specified in the subsequent subsections.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:product_data="urn:gs1:tsd:product_data:xsd:1"
xmlns:shared_common="urn:gs1:shared:shared_common:xsd:3"
xmlns:tsd_common="urn:gs1:tsd:tsd_common:xsd:1"
targetNamespace="urn:gs1:tsd:product_data:xsd:1" elementFormDefault="unqualified"
attributeFormDefault="unqualified" version="1.1">
  <xsd:annotation>
    <xsd:documentation><![CDATA[---------------------------
© Copyright GS1, 2015

GS1 is providing this XML Schema Definition file and resultant XML file as a service to
interested industries.
This XML Schema Definition file and resultant XML file were developed through a consensus
process of interested parties.

Although efforts have been made to ensure that the XML Schema Definition file and resultant XML
file are correct, reliable, and technically
accurate,  GS1 makes NO WARRANTY, EXPRESS OR IMPLIED, THAT THIS XML Schema Definition file and
resultant XML file ARE
CORRECT, WILL NOT REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE, OR
WILL BE SUITABLE FOR
ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR OTHERWISE.  Use of the XML Schema Definition
file and resultant XML
file are with the understanding that GS1 has no liability for any claim to the contrary, or for
any damage or loss of any kind or nature.

Version Information:
Version Number: 1.1
Date of creation: June 2013

The schema and subsequent updates will be provided on the GS1 websites.
---------------------------
]]></xsd:documentation>
  </xsd:annotation>
  <xsd:import namespace="urn:gs1:shared:shared_common:xsd:3"
schemaLocation="../shared/SharedCommon.xsd"/>
  <xsd:import namespace="urn:gs1:tsd:tsd_common:xsd:1" schemaLocation="TSDCommon.xsd"/>
  <xsd:complexType name="TSD_ProductComponentRecordType">
    <xsd:sequence>
      <xsd:element name="componentNumber" type="xsd:nonNegativeInteger"/>
      <xsd:element name="componentDescription" type="shared_common:Description1000Type"
minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="module" type="shared_common:ExtensionType" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="avpList" type="tsd_common:TSD_AttributeValuePairListType"
minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TSD_ProductDataRecordType">
    <xsd:sequence>
      <xsd:element name="productionVariantDescription" type="shared_common:Description70Type"
minOccurs="0"/>
      <xsd:element name="productionVariantEffectiveDateTime" type="xsd:dateTime"
minOccurs="0"/>
      <xsd:element name="module" type="shared_common:ExtensionType" maxOccurs="unbounded"/>
      <xsd:element name="productComponentRecord"
type="product_data:TSD_ProductComponentRecordType" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="avpList" type="tsd_common:TSD_AttributeValuePairListType"
minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TSD_ProductDataType">
    <xsd:sequence>
      <xsd:element name="gtin" type="shared_common:GTINType"/>
      <xsd:element name="targetMarket" type="shared_common:CountryCodeType"/>
      <xsd:element name="informationProviderGLN" type="shared_common:GLNType"/>
```

```
      <xsd:element name="informationProviderName" type="xsd:string"/>
      <xsd:element name="timeToLive" type="xsd:duration" minOccurs="0"/>
      <xsd:element name="productDataRecord" type="product_data:TSD_ProductDataRecordType"
maxOccurs="unbounded"/>
      <xsd:element name="avpList" type="tsd_common:TSD_AttributeValuePairListType"
minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

8.3 Interface Messages XML Schema

This section specifies XML schemas that implement request and response message payloads implied by the abstract definition of service interfaces specified in Section 7.

8.3.1 Aggregator-Aggregator Query Interface (AAQI) XML Schema

This section specifies an XML schema that implement the response message payloads implied by the abstract definition of the Aggregator-Aggregator Query Interface (AAQI) specified in Section 7.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:product_data="urn:gs1:tsd:product_data:xsd:1"
xmlns:query_by_gtin_response="urn:gs1:tsd:query_by_gtin_response:xsd:1"
xmlns:shared_common="urn:gs1:shared:shared_common:xsd:3"
xmlns:tsd_common="urn:gs1:tsd:tsd_common:xsd:1"
targetNamespace="urn:gs1:tsd:query_by_gtin_response:xsd:1" elementFormDefault="unqualified"
attributeFormDefault="unqualified" version="1.1">
  <xsd:annotation>
    <xsd:documentation><![CDATA[---------------------------
© Copyright GS1, 2015

GS1 is providing this XML Schema Definition file and resultant XML file as a service to
interested industries.
This XML Schema Definition file and resultant XML file were developed through a consensus
process of interested parties.

Although efforts have been made to ensure that the XML Schema Definition file and resultant XML
file are correct, reliable, and technically
accurate,  GS1 makes NO WARRANTY, EXPRESS OR IMPLIED, THAT THIS XML Schema Definition file and
resultant XML file ARE
CORRECT, WILL NOT REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE, OR
WILL BE SUITABLE FOR
ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR OTHERWISE.  Use of the XML Schema Definition
file and resultant XML
file are with the understanding that GS1 has no liability for any claim to the contrary, or for
any damage or loss of any kind or nature.

Version Information:
Version Number: 1.1
Date of creation: June 2013

The schema and subsequent updates will be provided on the GS1 websites.
---------------------------
]]></xsd:documentation>
  </xsd:annotation>
  <xsd:import namespace="urn:gs1:shared:shared_common:xsd:3"
schemaLocation="../shared/SharedCommon.xsd"/>
  <xsd:import namespace="urn:gs1:tsd:product_data:xsd:1" schemaLocation="ProductData.xsd"/>
  <xsd:import namespace="urn:gs1:tsd:tsd_common:xsd:1" schemaLocation="TSDCommon.xsd"/>
  <xsd:element name="queryByGtinResponse"
type="query_by_gtin_response:TSD_QueryByGTINResponseType"/>
  <xsd:complexType name="TSD_ImplementationExceptionType">
    <xsd:complexContent>
      <xsd:extension base="tsd_common:TSD_ExceptionType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_InvalidGTINExceptionType">
    <xsd:complexContent>
      <xsd:extension base="tsd_common:TSD_ExceptionType"/>
    </xsd:complexContent>
  </xsd:complexType>
```

```
  <xsd:complexType name="TSD_InvalidRequestExceptionType">
    <xsd:complexContent>
      <xsd:extension base="tsd_common:TSD_ExceptionType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_InvalidTargetMarketExceptionType">
    <xsd:complexContent>
      <xsd:extension base="tsd_common:TSD_ExceptionType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_NoDataExceptionType">
    <xsd:complexContent>
      <xsd:extension base="tsd_common:TSD_ExceptionType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_QueryByGTINResponseType">
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="productData" type="product_data:TSD_ProductDataType"/>
        <xsd:element name="noDataException"
type="query_by_gtin_response:TSD_NoDataExceptionType"/>
        <xsd:element name="invalidGTINException"
type="query_by_gtin_response:TSD_InvalidGTINExceptionType"/>
        <xsd:element name="invalidTargetMarketException"
type="query_by_gtin_response:TSD_InvalidTargetMarketExceptionType"/>
        <xsd:element name="unsupportedVersionException"
type="query_by_gtin_response:TSD_UnsupportedVersionExceptionType"/>
        <xsd:element name="invalidRequestException"
type="query_by_gtin_response:TSD_InvalidRequestExceptionType"/>
        <xsd:element name="securityException"
type="query_by_gtin_response:TSD_SecurityExceptionType"/>
        <xsd:element name="implementationException"
type="query_by_gtin_response:TSD_ImplementationExceptionType"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TSD_SecurityExceptionType">
    <xsd:complexContent>
      <xsd:extension base="tsd_common:TSD_ExceptionType">
        <xsd:sequence>
          <xsd:element name="exceptionContactDescription"
type="shared_common:Description200Type" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TSD_UnsupportedVersionExceptionType">
    <xsd:complexContent>
      <xsd:extension base="tsd_common:TSD_ExceptionType"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

9 Transport Bindings

This section defines concrete implementations of the service interfaces defined in Section 7 using particular transport technologies.

9.1 Web Service Bindings

This section defines a concrete implementation of the AAQI service interface (Section 7.1) using REST-style web services, where the response payloads are based on the XML schemas defined in Section 8. See Section 9.2 for a binding of the AIQI service interface (Section 7.2) based on ONS. In this version of the standard, no binding of the AIMI to a concrete service is specified.

9.1.1 REST Web Service Bindings

This section specifies an implementation of the AAQI service interface (Section 7.1) using web services defined in a Representation State Transfer (REST) style. This style is characterised by:

- HTTP is used as the transport protocol.
- The request is an HTTP GET request, where the request parameters are represented as URL query parameters.
- The response payload is the XML response document defined in Section 8
- HTTP response code 200 indicates success. HTTP response codes not beginning with "2" are used to indicate exceptions.

The specifics are defined in the following sections.

9.1.1.1 Protocol

In the REST web service binding for the AAQI the client of the interface SHALL send a request using HTTP, and the service SHALL respond using HTTP. Both the client and service SHALL conform to HTTP 1.1.

9.1.1.2 Security

In the REST web service binding for the AAQI the client and service SHALL use Transport Level Security (TLS) as defined in [RFC2818]. TLS for this purpose SHALL be implemented as defined in [RFC2246] except that the mandatory cipher suite is TLS_RSA_WITH_AES_128_CBC_SHA, as defined in [RFC3268] with CompressionMethod.null. Implementations MAY support additional cipher suites and compression algorithms as desired.

Mutual authentication of the AAQI client and service is provided through the use of message authentication codes as follows.

Prior to interaction via AAQI, data aggregators establish trust in the following manner. Data Aggregator A provides to Data Aggregator B a GLN that uniquely identifies Aggregator A as a TSD Data Aggregator and a symmetric key (as defined below) that Data Aggregator A will use to authenticate its identity to Data Aggregator B in the request and vice versa in the response. This key is referred to as $K_{A \to B}$. Data Aggregator B provides to Data Aggregator A the service URL that Data Aggregator B registers in the Global Index for GTINs for which B provides data. This exchange of information provides the ability for Data Aggregator A to issue requests to Aggregator B. As B will likely wish to issue requests to A the converse set of information is also exchanged, including a different key $K_{B \to A}$. Each symmetric key K SHALL be 256 bits in length.

When a client Data Aggregator A makes an AAQI request to a service Data Aggregator B, it SHALL include a Message Authentication Code (MAC) in the request URL calculated in the following manner:

8. Data Aggregator A constructs the request URL according to Section 9.1.1.3, including its GLN (as provided to Data Aggregator B during trust establishment) as the value of the `clientGln` parameter. At this stage, the request URL does not include the `mac` parameter.
9. Data Aggregator A computes a message authentication code using the HMAC algorithm [RFC2104] with SHA-256 [RFC4634] as the digest algorithm (a combination commonly referred to as HMAC-SHA256). The data input to the HMAC algorithm is the path and query portions of the request URL including the initial slash character and including all query parameters in the order chosen by Data Aggregator A. These URL characters shall be treated as octets in the US-ASCII encoding for purposes of this calculation. The key input to the HMAC algorithm is $K_{A \to B}$ as previously shared during trust establishment, which Data Aggregator A looks up as a function of B's service URL.

   For example, if the request URL (prior to adding the `mac` parameter) is the following:

   http://tsd.example.com/service/v1/ProductData/gtin/00614141123452?targetMarket=840&dataVersion=1.0&clientGln=0614141000005

   then the input to the HMAC algorithm are the following characters, encoded as US-ASCII octets:

```
/service/v1/ProductData/gtin/00614141123452?targetMarket=840&dataVersion=
1.0&clientGln=0614141000005
```

10. The resulting message authentication code is converted to a 64-character hexadecimal string (using uppercase letters) and appended to the request URL as the value of the `mac` parameter.
11. The final URL is used to make the HTTP request.

The server Data Aggregator B SHALL authenticate the identity of the client Data Aggregator A in the following manner:

12. Data Aggregator B receives the HTTP request.
13. Using the value of the `clientGln` parameter Data Aggregator B looks up the corresponding key $K_{A \to B}$. If no such key is known, stop: the client is unknown and a `SecurityException` is raised.
14. Data Aggregator B computes a message authentication code using HMAC-SHA256 where the data input to the HMAC algorithm is the path and query portions of the request URL including the initial slash character and including all query parameters except the `mac` parameter and where the key is $K_{A \to B}$.
15. Data Aggregator B compares the MAC computed in the previous step to the MAC included in the request URL's `mac` parameter. If the MACs are identical, then Data Aggregator A's identity is authenticated. Otherwise, stop: the client is not authenticated and a `SecurityException` is raised.

The service Data Aggregator does not authenticate to the client Data Aggregator prior to the request being delivered from client to server. Instead, the service Data Aggregator provides a MAC in the response so that the client may authenticate the server when it receives the response.

When a service Data Aggregator B responds to an AAQI request made by client Data Aggregator A, it SHALL include a Message Authentication Code (MAC) in the following manner:

16. Data Aggregator B computes a message authentication code using the HMAC-SHA256 where the data input to the HMAC algorithm is the entire HTTP response payload. The payload shall be treated as a sequence of bytes for the purposes of this calculation. The key input is $K_{A \to B}$ as previously shared during trust establishment, which Data Aggregator B looks up as a function of the `clientGln` parameter included in the request.
17. The resulting message authentication code is converted to a 64-character hexadecimal string (using uppercase letters) and included in the HTTP response as the value of an HTTP header whose name is `GS1-MAC`.

The client Data Aggregator A SHALL authenticate the identity of server Data Aggregator B in the following manner:

18. Data Aggregator A receives the HTTP response.
19. Data Aggregator A retrieves the key $K_{A \to B}$ previously used to create the request.
20. Data Aggregator A computes a message authentication code using HMAC-SHA256 where the data input to the HMAC algorithm is the response payload and where the key is $K_{A \to B}$.
21. Data Aggregator A compares the MAC computed in the previous step to the MAC included in the response's `GS1-MAC` HTTP header. If the MACs are identical, then Data Aggregator B's identity is authenticated. Otherwise, stop: the server is not authenticated and the response should not be trusted.

9.1.1.3 Request

A Data Aggregator acting as an AAQI client using the REST binding SHALL implement each interface method by formulating an HTTP request as specified in this section. A Data Aggregator acting as an AAQI server using the REST binding SHALL interpret an HTTP request as an invocation of an interface method as specified in this section and respond accordingly.

The HTTP URL for the request SHALL have the following form:

Base`/v1/`*Resource*`?`*Parameters*

where:

- *Base* is an absolute HTTP (or HTTPS) URL, determined as follows:
  - For the AAQI, *Base* is a URL obtained through the AAQI as the `dataAggregatorService` data element contained in an AIQI method response.
- The two characters `v1` denote that the operation being invoked is as specified in this version of the TSD standard. These characters are reserved to provide for a staged transition to an incompatible version of the TSD standard in the future. See Section 5.4.2.
- *Resource* is a URL segment as specified below according to the interface operation to be invoked by this request and the subject of that operation
- *Parameters* is an HTTP query string encoding parameters for the request beyond those encoded into *Resource*.

The HTTP method SHALL be GET and the HTTP request payload SHALL be empty.

The following table specifies the requests for the AAQI:

Interface operation	*Resource* portion of HTTP URL	HTTP Method	*Parameters* portion of HTTP URL	Request payload
`queryByGtin`	As specified below	GET	As specified below	Empty

The *Resource* portion of the HTTP URL for the AAQI `queryByGtin` operation SHALL have the following form:

`ProductData/gtin/`*`gtin`*

(Note that the second segment of the *Resource* portion is the four-character string `gtin` while the third segment is the 14-digit GTIN of the product.)

The *Parameters* portion of the HTTP URL for the AAQI `queryByGtin` operation SHALL have the following form:

`targetMarket=`*`targetMarket`*`&dataVersion=`*`version`*`&clientGln=`*`clientGln`*`&mac=`*`mac`*

where:

- *gtin* is the GTIN for the request (see Section 7.1.1).
- *targetMarket* is the target market code for the request (see Section 7.1.1).
- *dataVersion* is the desired data version for the request (see Section 7.1.1).
- *clientGln* is a 13-digit GLN that identifies the client; see Section 9.1.1.2.
- *mac* is a 64-character hexadecimal numeral (using uppercase letters) whose value is the message authentication code for the request as computed according to Section 9.1.1.2.

9.1.1.4 Response

A Data Aggregator acting as an AAQI server using the REST binding SHALL respond to an HTTP request as specified in this section. A Data Aggregator acting as an AAQI client using the REST binding SHALL interpret an HTTP response as specified in this section as a response to an interface operation.

A REST implementation of the AAQI SHALL process an incoming HTTP request as follows:

- If the HTTP URL matches the base URL and resource portion of the URL as specified in Section 9.1.1.3, then:
  - If the HTTP method (GET) matches the HTTP method specified in Section 9.1.1.3 for the operation named in the URL, then process the request as specified below.

- Otherwise, the HTTP method is inappropriate: respond with HTTP error code 405, with no HTTP response payload

- Otherwise, the request is outside the scope of this standard, and the response is implementation-defined. If the request is considered invalid by the implementation, the implementation SHOULD respond with HTTP error code 400 or 404.

Except as noted above, the response payload for the AAQI SHALL be an XML document conforming to the schema specified in Section 8.3.1 whose top level element is `queryByGtinResponse` and where the element contained immediately within the top level element is as specified in the table below. The response SHALL include an HTTP header named `GS1-MAC` whose value is a 64-character hexadecimal numeral (using uppercase letters) whose value is the message authentication code for the response as computed according to Section 9.1.1.2, except that the server MAY omit the MAC in the case of a `securityException` or `implementationException`. The server SHOULD include the message authentication code in the response for those exceptions if the client identity and its key are known to the server. The following table also specifies the HTTP response code for each outcome:

Outcome	HTTP response code	XML element contained within `queryByGtinResponse`
Success	200	`productData`
`NoDataException`	404	`noDataException`
`InvalidGtinException`	400	`invalidGtinException`
`InvalidTargetMarketException`	400	`invalidTargetMarketException`
`UnsupportedVersionException`	400	`unsupportedversionException`
`SecurityException`	403	`securityException`
`InvalidRequestException`	400	`invalidRequestException`
`ImplementationException`	500	`implementationException`

9.2 ONS 2.0 Binding for Aggregator-Index Query Interface (AIQI)

This section defines a concrete implementation of the Aggregator-Index Query Interface (AIQI) defined in Section 7.2 using ONS 2.0.

ONS 2.0 is an application of the Internet Domain Name System (DNS) which provides a means to look up a GS1 key and obtain a list of one or more services registered for that key. In the ONS binding for the AIQI, the GS1 key is the GTIN in the `queryByGtin` request, and the services registered in ONS are references to Data Aggregators that have data for that GTIN.

9.2.1 AIQI Query Via ONS

A Data Aggregator using the ONS 2.0 binding of the AIQI SHALL formulate a query by constructing the following Application Unique String (AUS), which [ONS_201] Section 6.1 defines as the input to the ONS 2.0 query process:

`|`*`country`*`|gtin|`nnnnnnnnnnnnn

where:

- *`country`* is replaced by the 2-character ISO 3166-1 alpha-2 country code [ISO3166] for the desired target market, converted to lowercase. Note that the 3-digit `CountryCode` as used in the abstract interface definition specified in Section 7.1.1 must be translated to the corresponding 2-character ISO 3166-1 alpha-2 code for the purposes of formulating the AUS. While ISO 3166-1 defines the alpha-2 code as uppercase, it must be converted to lowercase for this purpose.

- *nnnnnnnnnnnnnn* is replaced by the 14-digit GTIN for which product data is requested. If product data for a GTIN-8, GTIN-12, or GTIN-13 is requested, the Data Aggregator SHALL prepend as many zero ('0') digits as required to make 14 digits total in the AUS.

The Data Aggregator SHALL make a DNS query based on the AUS as specified in [ONS_201], Sections 6.2 and 7.1. If the Data Aggregator desires to include two or more target markets in the query, it may make a single DNS query because the country part of the AUS does not enter into the fully-qualified domain name (FQDN) computed from the AUS according to [ONS_201]. However, in matching the NAPTR records that are returned, the Data Aggregator SHALL formulate separate AUSs for each target market and treat each independently to match returned NAPTR records.

The response to the DNS query consists of zero or more DNS NAPTR records. The Data Aggregator SHALL apply the procedure specified in [ONS_201] Section 8 to obtain one or more service URLs for peer Data Aggregators. In applying this procedure, the Data Aggregator SHALL implement the procedure specified in [ONS_201] Section 8 as follows:

- In Step 2, only consider those NAPTR records whose Service field contains the following service type URL:

  `http://ons.gs1.org/tsd/servicetype/aaqi-1.xml`

- In Step 4, rather than attempting to find the first NAPTR record that matches the criteria, continue processing all NAPTR records and gather all service URLs that match the criteria. In this way, multiple peer Data Aggregators may be identified as a result.

9.2.2 NAPTR Records for TSD

A Global Index that implements the ONS 2.0 binding of AIQI must maintain DNS NAPTR records that will be interpreted properly by the procedures specified in Section 9.2.1.

Each NAPTR record defines the Data Aggregator registered for a particular combination of GTIN, target market, and language code. Nominally, each NAPTR record SHALL be associated to a DNS domain name resulting from transforming a GTIN according to [ONS_201], Section 6.2. If two or more GTINs have digits in common such that their corresponding DNS domain names share a common suffix, and those GTINs would have an identical set of NAPTR records, then the set of NAPTR records MAY be associated to a higher level DNS domain name using the DNS wildcard mechanism.

Each NAPTR record SHALL contain data as specified in [ONS_201], Section 7.2, and as noted below:

- The Flags field SHALL consist of the single character 'u', indicating that the Regexp field contains a URI.
- The Service field SHALL consist of the following service type URL:

  `http://ons.gs1.org/tsd/servicetype/aaqi-1.xml`

- The Regexp field SHALL be composed by concatenating the following strings:
  - The characters `!^\|` (exclamation point, caret, backslash, and vertical bar)
  - The two-letter ISO 3166-1 alpha-2 country code corresponding to the target market to which this NAPTR record applies, converted to lowercase. More complex regular expressions may be used to allow this NAPTR record to match more than one target market; see [ONS_201].
  - The characters `\|.*$!` (backslash, vertical bar, period, asterisk, dollar sign, and exclamation point)
  - The service URL for the Data Aggregator to which this NAPTR record refers
  - A `!` character (exclamation point)