GS1 Source

TSD v1.1 Technical Implementation Guide for Aggregators

Issue 1, Ratified, Jun-2014

Document Summary

Document Item	Current Value
Document Title	GS1 Source TSD v1.1 Technical Implementation Guide for Aggregators
Date Last Modified	Jun-2014
Document Issue	Issue 1
Document Status	Ratified
Document Description	TSD v1.1 Technical Implementation Guide for GS1 Source Aggregators

Contributors

Name	Organization
Dipan Anarkat	GS1 GO
Mark Frey	GS1 GO
Michael Dols	Met Labs

Log of Changes in Issue 1

Issue No.	Date of Change	Changed By	Summary of Change
Draft 1	3-Jan-2013	Dipan Anarkat	Updated the guide created for prequal certification with some additional content and to make it current with usage of TSD v1.1. - Changed references to TSD v1.1 from TSD v1.0, throughout the document - Updated sections ‘Audience’ & ‘Purpose’ - New sections added ‘TLS’ & ‘Content-Length’ - Added FAQ #7 on country codes. - New section on UTF-8 encoding added – needs discussion with group
Draft 2	9-Jan-2013	Dipan Anarkat	Updates made based on TSD MSWG group call on 9-Jan-2013. - General formatting changes and clean-up of document - Consistent usage of ISO terms of reference in the document - Guidelines updated for Byte Order Mark and Certificates.

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 Source TSD v1.1 Technical Implementation Guide for Aggregators** to agree to grant to GS1 members a royalty-free license or a RAND license 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 licensing obligations of GS1. Moreover, the agreement to grant licenses 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 organization is developing in compliance with the Specification and whether a license under a patent or other intellectual property right is needed. Such a determination of a need for licensing should be made in view of the details of the specific system designed by the organization 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.

Table of Contents

1. Introduction

1.1. Purpose

TSD v1.1 standards were developed in mid 2013 and since then, GS1 has launched multiple programs addressing compliance and certification to GS1 Source standards. To support GS1 Source conformance initiatives and to enable aggregator implementation, a comprehensive technical implementation guide is provided in this document. This implementation guide is based on the TSD v1.1 standard and provides important technical guidance for aggregator service implementation.

1.2. Audience

GS1 Source Data Aggregators – specifically developers and technical experts

1.3. References

- [TSD] GS1 Trusted Source of Data standard v. 1.1

  http://www.gs1.org/gsmp/kc/b2c

- [ONS] GS1 Object Name Service standard v. 2.0.1

  http://www.gs1.org/gsmp/kc/epcglobal/ons/ons_2_0_1-standard-20130131.pdf

1.4. Terms

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.

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.

2. Implementation Guidelines

2.1. HTTP Request

2.1.1. HTTP Request Parameters

Request Parameter	Implementation Guide
`gtin`	Value SHALL NOT be empty or a `null` value.
`targetMarket`	Value SHALL NOT be empty or a `null` value
`dataVersion`	Value SHALL be '1.1'
`clientGln`	Value SHALL NOT be empty or a `null` value.
`mac`	Value SHALL NOT be empty or a `null` value.

2.1.2. Use of HTTP vs. HTTPS protocol

The TSD specification requires TLS to be used for the AAQI web service binding. This requires the use of the HTTPS protocol, regardless if the URL in ONS had '`http`' in the aggregator web service URL. The URL returned via ONS should be used as the selector for the aggregator to be used and the AAQI request will always be made using the HTTPS protocol.

2.2. HTTP Response

2.2.1. HTTP Headers

It is important but not required to always label Web documents explicitly. Explicitly setting the appropriate HTTP headers is a good practice and will prevent the recipient from guessing the output.

2.2.1.1. GS1-MAC

The `GS1-MAC` header is used in the AAQI response and is included as a HTTP response header. It is calculated based on the payload that will be included in the AAQI response

The specification requires that you put a `GS1-MAC` header in the response every time, even in the case of exceptions. This header MAY be omitted in the case of a '`SecurityException'` and/or an '`ImplementationException`. This header SHALL be omitted in the event the identity (`clientGLN`) of the requesting aggregator cannot be confirmed. In this case, it would not be possible to calculate the mac code to put in the `GS1-MAC` header as the server would not know which symmetric key to use.

When available and included in the response, the value SHALL NOT be empty or a `null` value.

2.2.1.2. Content-Type

The value for media/MIME type in the HTTP entity header '`Content-Type`' SHOULD always be set appropriately to prevent applications from guessing it. When the XML payload contains a valid XML response, the media/MIME type SHOULD be set to '`application/xml`'.

In the case of HTTP errors when the response is implementation specified (HTTP error codes 400 or 404), the appropriate values for media/MIME type and ‘`charset`’ attribute SHOULD be included in the ‘`Content-Type`’ header (such as `text/plain`, `application/xhtml+xml`, etc.).

In the absence of ‘`charset`’ attribute in ‘`Content-Type`’, HTTP 1.1 standard (RFC 2616) specifies that the default character set is ISO-8859-1. This attribute MAY be omitted if the content of the HTTP response is encoded using the ISO-8859-1 character set. If the content is encoded using another character set (e.g. UTF-8 which is popular for XML content), then the ‘`charset`’ attribute SHALL be specified in the ‘`Content-Type`’ header and its value SHALL be set appropriately. Ensuring that the value of the ‘`charset`’ attribute matches the content encoding and declaring it explicitly when possible will prevent decoding errors on the HTTP client side.

Summarily, the media / MIME type declaration and the ‘`charset`’ attribute SHALL correspond to the actual media /MIME type and character encoding used for the content in the response.

2.2.1.3. Content-Length

The value for the HTTP entity header ‘`Content-Length`’ SHOULD be set to ‘`0`’ if the payload is empty and SHOULD be set appropriately if not empty. This guideline applies to the HTTP request as well.

2.2.2. Byte Order Mark

The Byte Order Mark (or BOM), is a special marker added at the very beginning of an Unicode file encoded in UTF-8, UTF-16 or UTF-32. It is used to indicate whether the file uses the big-endian or little-endian byte order. The BOM is mandatory for UTF-16 and UTF-32, but it is optional for UTF-8.

When the XML response is encoded using UTF-8 encoding, an optional BOM may be included in the response payload. The BOM affects the calculation of the mac code when computing the `GS1-MAC` header value. For consistent implementation and to prevent errors on the client side on whether the mac code was calculated using the BOM or not, it is important to specify if the BOM is included in the XML response.

The BOM SHALL be included if the encoding type is UTF-16 or UTF-32 and SHALL NOT be included in the XML response if the encoding type is UTF-8.

2.2.3. XML Content

When XML content is returned in the HTTP response, the XML prolog SHALL be declared and the appropriate value SHALL be specified for the ‘`encoding`’ attribute. The XML encoding type and the value of the ‘`encoding`’ SHALL match with the value of the ‘`charset`’ attribute of the ‘`Content-Type`’ HTTP response header. If this header is not declared in the HTTP response, the default encoding of ISO-8859-1 applies. The encoding used depends on the actual XML content and is not required to be UTF-8 as is commonly mistaken. Example –

```
<?xml version="1.0" encoding="UTF-8"?>
```

2.2.4. HTTP Error Codes

The table below provides information on specific error conditions and the HTTP error code to be returned in the HTTP response. It provides additional detail to the error conditions already specified in section 9.1.1.4 of the TSD specification.

Error Condition	HTTP Status / Error Code
Any HTTP method call besides GET (e.g. POST, DELETE, etc.)	405

2.2.5. AAQI Exceptions

The table below provides information on specific exception conditions and the AAQI Exception to be returned in the HTTP response. It provides additional detail to the exception conditions already specified in section 7.2 and 9.1.1.4 of the TSD specification.

Exception Condition	AAQI Exception
Parameter '`gtin`' not included in HTTP request	`InvalidRequestException`
Value of parameter '`gtin`' is empty (`null` value)	`InvalidRequestException`
GTIN is not 14 digits in length	`InvalidGTINException`
GTIN has a bad check digit	`InvalidGTINException`
Parameter '`targetMarket`' not included in HTTP request	`InvalidRequestException`
Value of parameter '`targetMarket`' is empty (`null` value)	`InvalidRequestException`
Parameter '`dataVersion`' not included in HTTP request	`InvalidRequestException`
Value of parameter '`dataVersion`' is empty (`null` value)	`InvalidRequestException`
Parameter '`clientGln`' not included in HTTP request	`InvalidRequestException`
Value of parameter '`clientGln`' is empty (`null` value)	`InvalidRequestException`
GLN is not 13 digits in length	`InvalidRequestException` or `SecurityException`
GLN has a bad check digit	`InvalidRequestException` or `SecurityException`
Parameter '`mac`' not included in HTTP request	`InvalidRequestException` or `SecurityException`
Value of parameter '`mac`' is empty (`null` value)	`InvalidRequestException` or `SecurityException`
Incorrect value for parameter '`mac`'	`SecurityException`
Unspecified / non-standard parameter specified in HTTP request	`InvalidRequestException`

2.3. Security

2.3.1. Certificates

The TSD specification requires the AAQI web service binding to support Transport Layer Security (TLS) with mandatory support for the `TLS_RSA_WITH_AES_128_CBC_SHA` cipher suite. It does not however prescribe the type of certificates to be used in the implementation of TLS, whether the certificate should be self-signed or signed by a Certificate Authority (CA). The SSL certificate to be used for TLS solves two purposes: encryption of traffic (for the RSA key exchange, at least) and verification of trust. There's a common misconception that self-signed certificates are inherently less secure than those sold by commercial CA's, that is incorrect. Self-signed certificates provide trust as well. If you securely distribute a self-signed certificate (or CA) and install it properly in the server, it's just as secure and is not vulnerable to man-in-the-middle attacks and certificate forgery. However, the difference is that commercial CA's provide extended verification of trust accomplished through a chain of certificates and by requiring more verification details from the requesting party.

For a production system certificates signed by a Certifying Authority SHOULD be used to add an additional layer of authentication and security. The type of signed certificates to be used should be bilaterally agreed upon between the aggregators. Although, aggregators MAY use self-signed certificates for beta deployments and testing purposes, its use in a production system is not recommended.

If using self-signed certificates, there is no need to validate the certificate path leading to a CA. Certificate path validation should be disabled, implying all certificates should be trusted. This setting is dependent on the HTTPS client used for application development. The peer aggregator certificate would have to be added to the certificate store / trust pool for validation to succeed.

Note that although a CA certificate may have been used, TSD uses mac code validation which includes GLN of the aggregator to authenticate its identity.

2.3.2. TLS

TLS version to be used SHALL be 1.1 or later. TLS 1.0 is now deprecated due to serious security vulnerability in TLS 1.0 when combined with SSL 3.0. Server handshake failure is likely to be encountered if an aggregator is trying to use TLS 1.0 and the peer aggregator requires higher version of TLS to be used. Check server configuration and update systems if required to ensure that TLS 1.1 or higher is in use. Some older systems may have TLS 1.0 as default and this may cause problems.

2.4. ONS Client Implementation

To implement an ONS client, refer to [ONS] and [TSD] specifications. The ONS specification specifies how to formulate the DNS queries to query information in the Federated ONS network. The TSD specification additionally provides information on how to formulate AIQI query for ONS 2.0.

The data aggregator must create an ONS client software that will issue a DNS query to an ONS peer root server and get a DNS NAPTR record back. The NAPTR record will contain the aggregator service URL information that would then be parsed.

The process would be as follows -

1. Convert GTIN to ONS 2.0 AUS
2. Transpose AUS to a ONS 2.0 FQDN
3. Issue a DNS query to DNS subsystem for a NAPTR record
4. Receive DNS NAPTR record(s)
5. Filter and fetch a single NAPTR record corresponding to the AUS, by applying ONS processing rules to NAPTR record fields (Order, Pref, Flags, Service and Regexp). An approach to select the matching NAPTR record is provided below. Implementations should choose their own logic on as they see fit. See section 7.2 of the ONS 2.0.1 specification for more details on the NAPTR resource record.
    a. Filter and fetch NAPTR record(s) matching AAQI service URN in the ‘Service’ field
    b. Filter and fetch NAPTR record(s) matching target market country code in the AUS by applying ONS processing rules to ‘Order’ and ‘Regexp’ field
    c. Filter and fetch a single NAPTR record having lower value in the ‘Pref’ field
6. Parse ‘Regexp’ field of NAPTR record to retrieve aggregator service URL

Once the technical specifications are understood, this is a fairly straightforward process.

3. Examples

The examples in this section have been provided for the purpose of illustration of the concept and are not real.

3.1. AAQI HTTP Request Response

Request URL:

```
http://tsd.example.com/v1/ProductData/gtin/43215678?targetMarket=USA&dataVersion=1.1&clientGln
=9506000038100&mac=null
```

Request Method: GET

Status Code: 403 Unauthorized

Request Headers:

```
GET
/v1/ProductData/gtin/43215678?targetMarket=USA&dataVersion=1.1&clientGln=9506000038100&mac=nul
l
HTTP/1.1
Host: tsd.example.com
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-US,en;
```

Query String Parameters:

```
targetMarket=USA&dataVersion=1.1&clientGln=9506000038100&mac=null
```

Response Headers:

```
HTTP/1.1
403 Unauthorized
Date: Thu, 20 Jun 2013 19:23:59 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: application/xml; charset=utf-8
GS1-MAC: 78C6FCA3C3421B7687FF4ABFCC045E2F8D6F36343B8257BB9821B622B2AE8B8E
Content-Length: 611
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

Response Body:

```
<?xml version="1.0" encoding="utf-8"?>
<query_by_gtin_response:queryByGtinResponse
xmlns:query_by_gtin_response="urn:gs1:tsd:query_by_gtin_response:xsd:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:gs1:tsd:query_by_gtin_response:xsd:1 QueryByGtinResponse.xsd
urn:gs1:tsd:basic_product_information_module:xsd:1 BasicProductInformationModule.xsd
urn:gs1:tsd:nutritional_product_information_module:xsd:1
NutritionalProductInformationModule.xsd">
      <securityException>
            <exceptionReason>Invalid Signature</exceptionReason>
      </securityException>
</query_by_gtin_response:queryByGtinResponse>
```

3.2. Resolution of GTIN to a AAQI service URL using ONS 2.0

GTIN: 9506000036908

Target Market: Belgium

ONS Peer Root: ons.example.com

Step	Activity	Actor	Format	Example Data
1	Convert GTIN to ONS 2.0 AUS	AAQI service	ONS 2.0 AUS	`|be|gtin|09506000036908` ▪ Note the addition of the digit ‘0’ to the beginning to pad the GTIN to a full 14 digits. ▪ The lower case ISO 3166-1 alpha-2 country code for target market ‘Belgium’ is included in the AUS
2	Transpose AUS to a ONS 2.0 domain name	AAQI service	DNS FQDN (Fully Qualified Domain Name)	`0.0.9.6.3.0.0.0.0.6.0.5.9.gtin.gs1.id.ons.example.com.`
3	Issue DNS query for a NAPTR (Name Authority Pointer) record	ONS Resolver (ONS client handling interaction with DNS subsystem)	DNS query for ‘NAPTR’ record	The format of the query is implementation dependant. As an example, one would pass the domain name for which the NAPTR record is required – `dns://0.0.9.6.3.0.0.0.0.6.0.5.9.gtin.gs1.id.ons.example.com. “NAPTR”`

<table>
<tr><th>Step</th><th>Activity</th><th>Actor</th><th>Format</th><th>Example Data</th></tr>
<tr><td>4</td><td>Receive DNS NAPTR record(s)</td><td>ONS Resolver</td><td>DNS “NAPTR” Record(s)</td><td>In this example, the authoritative name server returns four NAPTR records available for the same GTIN
1. NAPTR for target market ‘be’ and primary AAQI service (pref=0)
NAPTR 0 0 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://be.tsd.example.com/! .
2. NAPTR for target market ‘be’ and backup AAQI service on European server (pref=1)
NAPTR 0 1 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://eu.tsd.example.com/! .
3. NAPTR for AAQI service in target market ‘us’
NAPTR 1 0 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|us\|.*$!http://us.tsd.example.com/! .
4. NAPTR for EPCIS service
NAPTR 0 0 “u” “EPC+ws” !^.*$!http://epcis.example.com/! .</td></tr>
<tr><td>5</td><td>Filter and fetch a single NAPTR record corresponding to the AUS, by applying ONS processing rules to NAPTR record fields</td><td>AAQI service</td><td>DNS “NAPTR” Record(s)</td><td>NAPTR 0 0 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://be.tsd.example.com/! .</td></tr>
</table>

<table>
<tr><th>Step</th><th>Activity</th><th>Actor</th><th>Format</th><th>Example Data</th></tr>
<tr><td>*5a*</td><td>*Filter and fetch NAPTR record(s) matching AAQI service URN in the ‘Service’ field*</td><td>*AAQI service*</td><td>*DNS “NAPTR” Record(s)*</td><td>NAPTR 0 0 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://be.tsd.example.com/! .
NAPTR 0 1 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://eu.tsd.example.com/! .
NAPTR 1 0 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|us\|.*$!http://us.tsd.example.com/! .</td></tr>
<tr><td>*5b*</td><td>*Filter and fetch NAPTR record(s) matching target market country code in the AUS by applying ONS processing rules to ‘Order’ and ‘Regexp’ field*</td><td>*AAQI service*</td><td>*DNS “NAPTR” Record(s)*</td><td>NAPTR 0 0 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://be.tsd.example.com/! .
NAPTR 0 1 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://eu.tsd.example.com/! .</td></tr>
<tr><td>*5c*</td><td>*Filter and fetch NAPTR record having lower value in the ‘pref’ field of NAPTR record*</td><td>*AAQI service*</td><td>*DNS “NAPTR” Record(s)*</td><td>NAPTR 0 0 “u” “http://ons.gs1.org/tsd/servicetype/aaqi-1.xml” !^\|be\|.*$!http://be.tsd.example.com/! .</td></tr>
<tr><td>6</td><td>Parse ‘Regexp’ field of NAPTR record to retrieve aggregator service URL</td><td>AAQI service</td><td>URL</td><td>http://be.tsd.example.com/</td></tr>
</table>

4. FAQ

1. **I do not have a GLN for my aggregator. Will GS1 provide me with a GLN?**

   You will need to assign a GLN to your aggregator service that will be provided to all other aggregators you communicate with. Obtaining a GLN is the aggregator's responsibility and can be obtained by contacting your local GS1 member organization.

2. **Standard TSD 1.1 schema definitions are assuming that the files are obtained from the file system, is the directory structure going to be maintained?**

   Currently, the schema files are distributed as a downloadable archive for local storage and import into applications. In the future, GS1 may also make the files available online in a permanent schema repository. The structure would be identical on the file system and online when available.

3. **The version parameter in the AAQI request is not very explicit. The XML schema regulates that it is a string from 1 to 35 characters. There is an example of the version as "v1.1". Is this the rule to follow? Will there be a fixed rule? Can anyone use their own strings for version definition ("v1.1", "version 1.1", "ver1.1", "1.1", etc)? If yes, how can we validate a request?**

   The version parameter is the desired version of the `ProductData` module. Implicitly, the version number of `ProductData` module is the same as version number of the TSD standard and that is '1.1'.

4. **To make an AIQI / ONS 2.0 and AAQI query, the GTIN needs to be in GTIN-14 format. However, the GTINs available are not in GTIN-14 format. What do I do?**

   GTINs have various sizes: GTIN-8, GTIN-12, GTIN-13 & GTIN-14. No matter which GTIN you get you always convert it to a GTIN-14 by zero padding it on the left.

5. **The aggregator service URL returned by the ONS has the protocol identifier as 'http'. Shouldn't this be 'https'?**

   It is undefined in the TSD specification. It is possible that an aggregator might chose to have a web page that provides the web service description. The AAQI requests will always be over 'https' as the specification mandates the use of TLS, so it does not matter if the URL returned by the ONS framework has 'http'.

6. **What URL do we use for the original AIQI call, as I do not see it in the documentation?**

   There is no URL to call. The AIQI is an abstract interface. For TSD v1.1, a technical binding to ONS 2.0 has been provided in the specification. ONS is a DNS application and as such the AIQI query is nothing but a standard DNS query.

7. **Country codes in the ONS system are listed as ISO Alpha 2 country codes, but the aggregator service only accepts three digit numeric country codes. Why is it not listed in the ONS as three-digit numeric?**

   The reason why there is a change in the representation format for country code is because the ONS system was created prior to and independent of any application like TSD. For country codes the ISO 2-character *'ISO 3166-1 alpha-2'* 2 character country code in uppercase was used in ONS. Now, the TSD Aggregator service was designed to use a 3 digit numeric country code largely due to the requirement to maintain full interoperability with GDSN, which uses *'ISO 3166-1 numeric-3'* 3 digit numeric codes. During the standards development, the working group was aware that a translation would be required and chose to keep the current design to maintain more full interoperability with GDSN.