GS1 Object Name Service (ONS)
Version 2.0.1

Ratified Standard
Issue 2 January 31, 2013

Document Summary

Document Item	Current Value
Document Title	Draft Standard Specification GS1 Object Name Service (ONS)
Date Last Modified	January 31, 2013
Document Issue	Issue 2 of Ratified Standard-updated with fixed errata
Document Status	Ratified Standard
Document Description	

Contributors

Name (& Notable Role)	Organization
Sandoche Balakrichenan-- Conformance Requirements Editor and Prototype Test Participant	AFNIC
Mark Harrison	Auto-ID Labs
Philippe Gautier	Business2Any
Marc-Antoine Mouilleron	France Telecom Orange
Philippe Rodier	France Telecom Orange
Jacques Madelaine	GREYC
Adrien Laurence	GREYC
Jerome Le Moulec	GREYC
Steven Pereira	GS1 Australia
Manfred Piller	GS1 Austria
Kevin Dean Co—Chair, Editor of ONS Solution; GS1 AG Liaison	GS1 Canada
Han Du	GS1 China
Zhang Xu	GS1 China
Giovanni Biffi	GS1 Colombia
Juan Ochoa	GS1 Colombia
Felipe Serrano	GS1 Columbia
Douglas Hill	GS1 Denmark
Per Kiilsholm	GS1 Denmark
Pertti Hakala	GS1 Finland
Nicolas Pauvre co-Chair, Prototype Test Participant	GS1 France
Andreas Fuessler	GS1 Germany
Ralph Troeger	GS1 Germany
Dipan Anarkat	GS1 Global Office

Name (& Notable Role)	Organization
Henri Barthel	GS1 Global Office
Mark Frey-- Facilitator & Process Manager	GS1 Global Office
Janice Kite	GS1 Global Office
Sean Lockhead	GS1 Global Office
Craig Alan Repec	GS1 Global Office
KK Suen	GS1 HK
Tany Hui --Prototype Test Participant	GS1 Hong Kong
Albert Tsang	GS1 Hong Kong
Noriyuki Mama	GS1 Japan
Reiko Moritani	GS1 Japan
Martin Beno	GS1 Slovakia
Miroslava Staffenova	GS1 Slovakia
Jeremy Morton--Prototype Test Participant	GS1 Sweden
Steffen Olsson	GS1 Sweden
Heinz Graf	GS1 Switzerland
Francis Kienlen-- Prototype Test Participant	GS1 Switzerland
Daniel Mueller	GS1 Switzerland
Christian Schneider	GS1 Switzerland
Andrew Osborne	GS1 UK
Roberto Quilez	INRIA
Ken Traub	Ken Traub Consulting LLC

Log of Changes in Ratified Standard

Issue No.	Date of Change	Changed By	Summary of Change

Disclaimer

Table of Contents

1. Introduction

This document gives an overview of Object Name Service (ONS) and specifies how ONS uses the Domain Name System (DNS) infrastructure and implementation. This document is a formal specification of an implementation of the Dynamic Delegation Discovery System (DDDS) algorithm to locate authoritative metadata and services associated with a given GS1 Identification Key. Its target audience is all interested parties who seek to have a technical overview of ONS and particularly to developers that will be implementing ONS resolution systems for applications.

1.1. Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest version of this document series is maintained by GS1. This document is the ratified version of the GS1 Object Name Service (ONS) and has been updated to fix some errata

Comments on this document should be sent to GSMP@gs1.org.

2. Terminology and Typographical Conventions

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 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.

3. ONS Introduction (non-normative)

In keeping with the assumption that GS1 standards should leverage existing standards and infrastructure where appropriate, ONS uses the Internet's existing DNS for looking up (resolving) information about a GS1 Identification Key. This means that the query and response formats must adhere to the DNS standards, meaning that the GS1 Identification Key will be converted to a domain name and the results must be a valid DNS Resource Record (RR).

Important terminology note: the usage of the terms "ONS" and "DNS" may seem arbitrary but they are not. The term DNS is used when the discussion is generally applicable to the DNS system. ONS is used when the discussion is specifically about querying the DNS for a GS1 Identification Key. For example, a query for a mail exchange (MX) resource record (RR) is a DNS query. A query to locate an EPCIS server through a GS1 Identification Key would be called an ONS query, even though the query is carried out using DNS.

Additionally, it is important to outline the difference between a "service" and a "server" in the context of this document. A service is a set of functions that accomplish some task. That set of functions may actually be implemented using one or more networked computer systems (which we refer to as servers) acting in concert. A good example is the concept of a "Local ONS", or what is commonly

referred to as a local name service, implementing ONS functionality. In most situations this service is provided by two physically separate servers that act as backups for one another.

3.1. Background Information

This document draws from the initial work at the Auto-ID Center, and we recognize the contribution of the following individuals: Joe Foley (MIT), Erik Nygren (MIT), Sanjay Sarma (MIT), David Brock (MIT), Sunny Siu (MIT), Laxmiprasad Putta (OATSystems), Sridhar Ramachandran (OATSystems). The following papers capture the contributions of these individuals:

- Engels, D., Foley, J., Waldrop, J., Sarma, S. and Brock, D., "The Networked Physical World: An Automated Identification Architecture" Proceedings of the 2nd IEEE Workshop on Internet Applications (WIAPP '01), 76-77, 2001.
- The Object Name Service Technical Manual, Version 0.5 (Beta) http://www.autoidlabs.org/whitepapers/MIT-AUTOID-TM-004.pdf

3.2. The Domain Name System (DNS)

This section is for readers who may not be familiar with DNS. Explaining the DNS infrastructure and implementation is not within the scope of this document. Hence, this is a basic introduction only and for those interested, references to appropriate technical documents on DNS will be provided.

In order to have a basic understanding of DNS it is necessary to understand the system from two major viewpoints. The first is from the viewpoint of the client that is querying DNS. The second is from the viewpoint of an entity publishing data into DNS to be queried by a client.

3.2.1. Client's View

A classical DNS usage pattern is to resolve the IP address of a Fully Qualified Domain name (FQDN) such as "www.gs1.org". A user who wishes to view the web site "www.gs1.org" uses an application such as a web browser. The application sends a request to a stub resolver in the local operating system. The stub resolver will invariably have a cache containing the results of recent DNS requests. If the cache can provide the answer to the request, the resolver will return the value in the cache to the application that made the request. If the cache does not contain the answer, the resolver will send the request to one or more designated DNS servers until it obtains the response for the request. Each of these DNS servers may in turn delegate to other DNS servers further up the chain.

From the application (e.g. web browser) standpoint, the DNS is a black box. Questions go in and answers come out. Where the answer comes from, how long it can be cached, and what sequence of delegations it took to find the answer are all hidden from the application that issues the query.

A DNS resolver used to perform a query can resolve the query with a minimum of two pieces information: the domain name in question and the type of resource record (RR) that is being requested.

For example, on a computer that has a properly configured network, the following snippet of Java code is one of the many possible ways to issue a DNS query and retrieve the results:

```
import javax.naming.directory.*;
DirContext ictx = new InitialDirContext();
Attributes attrs = ictx.getAttributes("dns://corp.example.com", "MX");
```

The "corp.example.com" is the domain name and the "MX" is the record type that is being requested. "MX" is the "mail exchanger" that specifies a server responsible for accepting email for a domain. When the DNS resolution process is successful, the "attrs" variable will contain the hostname for the MX server for the "corp.example.com" domain. All other DNS issues such as finding the root servers

and the authoritative name servers, how long the DNS record is stored in the cache etc., are hidden from the application.

For a more detailed explanation of how the DNS protocol works please refer to [DNS].

3.2.2. Publisher's View

While applications are happily able to ignore how the DNS infrastructure actually works, the data the client is consuming must be provisioned in a way that is understandable by the DNS protocol. A basic introduction to the key concepts of DNS infrastructure is provided here.

DNS is a distributed database that is indexed by domain names. Each domain name is essentially just a path in a large inverted tree called the domain namespace. The tree has a single root at the top. It is referred simply as the "root" and denoted by a "." (dot). Entries in the root "zone" are called Top Level Domains (TLDs) such as ".com", ".net", ",org", ".fr", ".us", etc. The process of separating a descendant of a zone into a separate zone is called "*delegation*". The delegation is accomplished with Nameserver (NS) records. The delegation system enables hosts to control given chunks of the database and the whole database is reachable by traversing the path from the root to each lower level; each point at which there is a "." in a domain name, there is a delegation to domains lower in the hierarchy. Generally speaking, for each delegation there is a corresponding network server that contains data for that subsection of the hierarchy. In the previous example of "corp.example.com" the delegation of servers would look as in Figure 1:

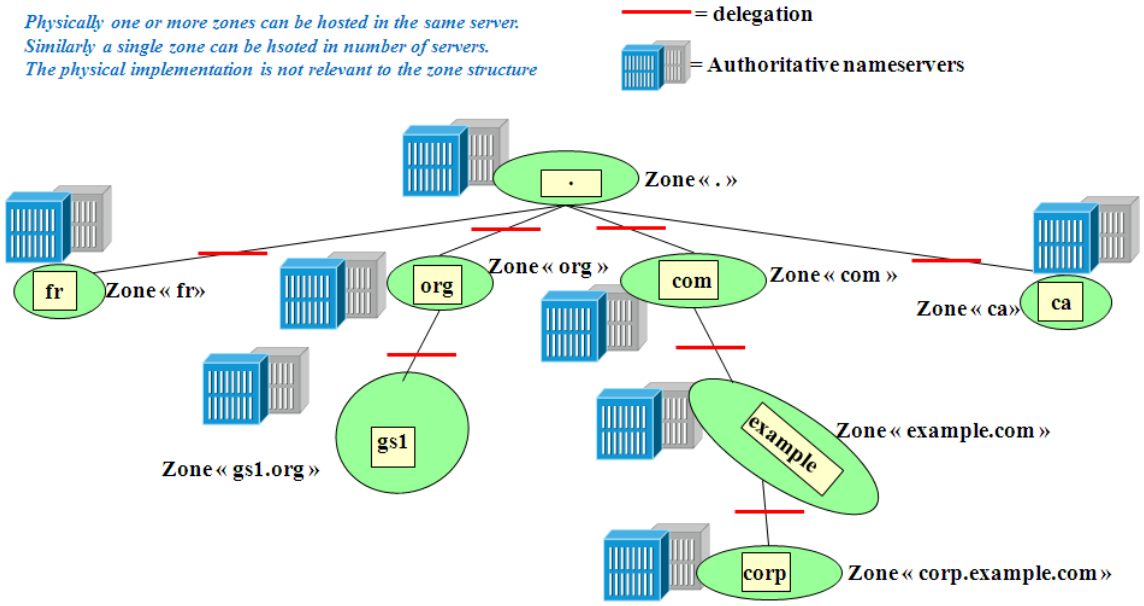


Figure 1 – Server delegation

The server that stores information about a domain namespace is called a nameserver. For example the root domain has thirteen nameservers, each of which contains the same data. The data that a nameserver publishes is called a "zone". In many situations a single nameserver can contain multiple zones.

Nameservers generally have complete information about some part of the domain namespace. There are two types of nameservers, Authoritative and Caching.

The Authoritative nameserver contains an entire copy of the zone that is derived from the local configuration data, possibly with the help of another authoritative nameserver for the zone. A server can be authoritative for one zone but not authoritative for another. The Master (primary) server is an Authoritative nameserver that gets its zone data from the local configuration, not from an outside

source. This is where all changes to a zone's contents are made. The DNS protocol provides an automatic mechanism for propagating the contents of a zone to slave (secondary) servers, which will be queried in case of the failure of primary servers.

A caching nameserver is usually a local nameserver. It initially looks up a name within the current zone and ends up asking one of the zone's nameservers for the answer and keeps the result of all the name resolutions in its cache for a Time To Live (TTL). The TTL on a resource record is the length of the time for which any nameserver can cache that record.

The data associated with domain names are contained in Resource Records (RRs). These define some attribute for a domain name such as its IP address, a string of text, or a mail route. A nameserver RR declares that a given zone is served by a given nameserver. Every nameserver record is either a delegation record or an authority record. If the name of the nameserver record is the name of the zone it appears in, it is an authority record. If the name of the nameserver record is that of a descendant zone, then it is a delegation record.

For further details please refer to [DNS].

3.3. ONS's Usage of DNS

In order to use DNS to find information about an item, the item's GS1 Identification Key must be converted into a format that DNS can understand, which is the typical, "dot" delimited, left-to-right form of all domain names.

As the purpose of ONS is to discover data and services associated with a GS1 Identification Key and multiple sets of data and services may exist for that key, the appropriate DNS record type is the Naming Authority PoinTeR (NAPTR) [RFC 3403]. This record type contains several fields for denoting the protocol, services, and features that a given service endpoint exposes. It also allows the service endpoint to be expressed as a URI, thus allowing complex services to be encoded in a standard way.

Figure 2 describes a typical ONS query from start to finish from the viewpoint of an application. In this example, the starting point is a bar code or RFID tag. However, the source of the GS1 Identification Key is not restricted to data carriers; it could be part of a transaction document (e.g. a purchase order), an event record, a master data record, or any other source.

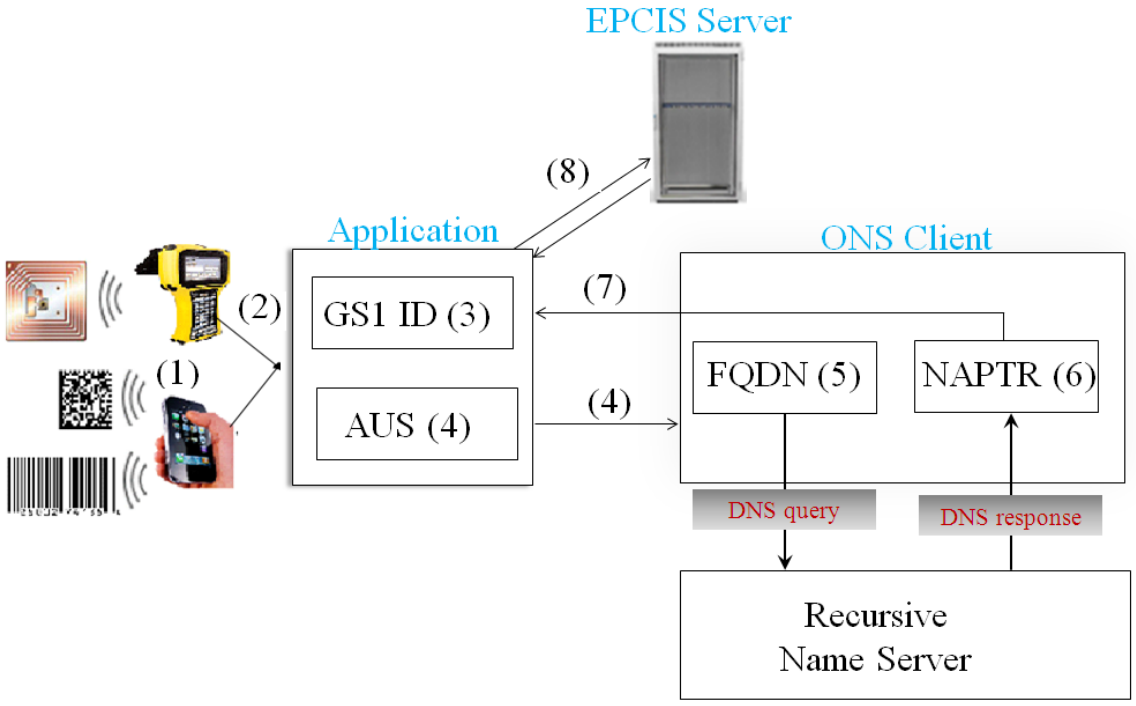


Figure 2 – A typical ONS query

1. A sequence of data denoting a GS1 Identification Key and optional supplemental data (e.g. ship-to location, batch/lot number, etc.) is read from a bar code or an RFID tag using an appropriate reader. When presented to the application layer, this data will be in text form.
2. The reader sends that sequence of data to an ONS application.
3. The ONS application extracts the desired GS1 Identification Key type and the GS1 Identification Key itself from the sequence of data. It should be noted that there is no requirement that the desired GS1 Identification Key be the primary GS1 Identification Key in the data stream. For example, while a shipping container will carry a Serial Shipping Container Code (SSCC) as a primary identifier, the application may be interested in discovering services associated with the Global Trade Item Number (GTIN) in the container. The GTIN is indicated using AI 02 in a supplementary bar code accompanying the SSCC. For example, if the data extracted from the bar codes, converted to a GS1 Element String, is as follows:

   `(00)306141417782246356(02)50614141322607(37)20`

   then the contained GTIN is `50614141322607`.
4. The ONS application presents the GS1 Identification Key type, the GS1 Identification Key itself, the client language code (optional), and the client country code (optional) to the ONS client. For example, assuming an application, configured to operate in English in Canada:

   `en|ca|gtin|50614141322607`
5. The ONS client converts the GS1 Identification Key type and the GS1 Identification Key into an appropriate FQDN as per Section 6.2 and issues a DNS query for NAPTR records for that domain. Example:

   `5.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.onsepc.com`

6. The DNS infrastructure returns a series of answers that contain service types and associated data (likely Uniform Resource Locators (URLs) that point to one or more services such as EPCIS or Mobile Commerce). See Section 9 for examples.
7. The ONS client extracts the service type and the service data from the DNS NAPTR record, interprets it according to the rules in Section 8 and presents it back to the ONS application.
8. The application interprets the data appropriate to the service type. Example:

   The application contacts the EPCIS server at the resolved URL.

3.3.1. Serial Number Level Queries to the ONS

It is important to note that this version of ONS does not support queries at the serial number or "instance" level. Instead, the lowest level of granularity supported is the "Object Class" level. Subsequent queries for information about a given serial number must be resolved by querying the application layer server designated by the ONS results. This is true for all GS1 Identification Keys. The ability to specify an ONS query at the serial number level as well as the architectural and economic impacts of that capability is an open issue that may be addressed in subsequent versions of this document. Its lack of mention here should not be construed as making that behaviour legal or illegal.

3.4. Why the Federated Model?

The ONS lookup function, being based upon DNS, is implemented by a distributed set of servers with a single (replicated) root, currently at onsepc.com.

The governance model and the architecture of the ONS have therefore raised concerns within some industrial and political communities. These concerns were notably expressed in the Internet of Things action plan for Europe[1].

The ONS design is consequently considered as a political issue that has to be negotiated on a governance perspective before it can be implemented. In particular, logistic applications that could imply sensitive product localization (e.g. drugs, weapons, nuclear waste…) or sensitive applications (product recall, anti-counterfeiting applications, food or drugs safety control…) are considered as critical applications as well as the architectures that support them. Many countries have indeed laws and regulations forbidding hosting of critical citizen and industry data outside their borders. Initial country routing (via onsepc.com) violates that requirement.

Concisely, in the global implementation of ONS, it has become thus evident that having a single resolution root for the entire system is not acceptable, due to the following concerns:

- Political concerns (risk of loss of sovereignty, control over information may be lost when subject to laws/control of other countries, etc.)
- Capability concerns (risk of loss of domestic and strategic capability (risk of loss of infrastructure in case of breakdowns, risk of service level reduction compared with local suppliers, etc.)
- Security concerns (risk of business intelligence gathering, etc.)
- Commercial concerns (risk of high price and maintenance costs, risk of financial viability of the supplier, risk of contract lock-in, risks arising from repatriation and/or transfer to another supplier if required, etc.)
- Innovation concern (no competitive challenge for technical innovation & marketing, etc.)

[1] http://ec.europa.eu/information_society/policy/rfid/documents/commiot2009.pdf

The answer to these issues is that ONS shall support a federated model in which multiple ONS Peer Roots may co-exist. Each ONS Peer Root shall be equivalent in functionality to every other; queries at any peer shall resolve to the same results regardless of the peer initially chosen. The ONS Peer Roots shall work collaboratively at the same hierarchical level to resolve client queries while respecting each peer's scope of authority.

This federated ONS model will be thus better adapted to local ownership and control requirements and will mitigate risk of single-country control.

From this point forward, the term "peer" shall be taken to refer to any one of the equivalent ONS Peer Root nodes within the federation.

3.5. Actors

A number of actors are referred to throughout this document. These may be legal entities, persons, or automated systems.

GS1 Global Office – This is the office located in Brussels, Belgium. It is not a traditional head office in that its work is driven by the country-level GS1 Member Organizations but it has a coordinating responsibility among all the GS1 country offices.

GS1 Member Organization – This is an office with responsibility for a country (e.g. GS1 Canada) or set of countries (e.g. GS1 UK).

GS1 Company Prefix Subscriber – This is an organization that has licensed a GS1 Company Prefix from a GS1 Member Organization. The traditional term "brand owner" is often used, but as a brand refers to a product and ONS supports all GS1 Identification Keys, the term GS1 Company Prefix Subscriber is preferred.

User – This is a human using an application to interact, directly or indirectly, with ONS.

Application – This is a traditional computer software application.

Client – This is a set of software libraries that may be part of an application or may be remotely accessible to an application. The application provides its parameters (the Application Unique String (AUS) and optionally the service(s) it is looking for) to the client. The client converts the AUS to a Fully Qualified Domain Name (FQDN) and is capable of performing a recursive search of the ONS infrastructure to locate Resource Records (RRs) required by the application.

ONS Peer Root – This is a logical server (optionally composed of multiple physical distributed servers), partitioned by the range of GS1 Company Prefixes that it supports. For queries within the range of GS1 Company Prefixes that it supports, it either answers them directly or delegates to another name server in the sub-domain; for queries outside the range of GS1 Company Prefixes it supports, it delegates to another ONS Peer Root.

4. ONS Delegation Architecture Organization

The previous section covered DNS and how ONS uses DNS from a client point of view. What it did not cover was the processes and procedures for making ONS data available in the proper form for the client. The main issue discussed in this section will be delegation for ONS at different levels.

Throughout this section, a few sample domains will be used. These are samples only and do not authoritatively identify the domains that may be chosen by the Member Organizations (MOs) that implement distributed ONS nodes. The domains are:

Domain	Usage
onsepc.com	ONS maintained by GS1 Global
ons.epcglobalcanada.org	Canadian ONS

Domain	Usage
onsepc.fr	French ONS
ons.epcglobal.cn	Chinese ONS

For purposes of example, the following GTIN and serial number in GS1 Element String form will be used:

```
(01)50614141322607(21)400
```

wherein "01" is the GTIN Application Identifier (AI), 50614141322607 represents the GTIN, "21" is the serial number AI, and 400 is the serial number.

Other GS1 identification keys have similar structure. Since the GTIN is the one most familiar to readers, it is used throughout.

Note again that the serial number in all implementations is actually irrelevant to ONS as ONS works only at the class level.

4.1. Delegation

An ONS Peer Root is one that may be used as a starting point for any ONS query. All ONS Peer Roots SHALL resolve to the same set of results.

At the ONS Peer Root level there are two types of delegations:

1. Nameserver (NS) and
2. Delegation name (DNAME).

NS delegation preserves the domain name and simply partitions a portion of the space to another nameserver. DNAME delegation, however, replaces one domain name with another and continues the query on the other domain name.

All GS1 Company Prefix Subscribers that wish to manage their own ONS entries under an ONS Peer Root namespace SHALL use NS delegation. An example of a NS delegation for a GS1 Company Prefix Subscriber at ONS Peer Root "onsepc.com" is:

```
1.4.1.4.1.6.0.gtin.gs1.id.onsepc.com. IN NS ns1.corp.example.com.
```

```
1.4.1.4.1.6.0.gtin.gs1.id.onsepc.com. IN NS ns2.corp.example.com.
```

Assuming that GS1 US uses "onsepc.com" as its root, a user under the GS1 US namespace, would query using the FQDN:

```
5.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.onsepc.com
```

As onsepc.com is the authoritative domain for GS1 US, this query would be resolved locally (with the caveat that there might be further delegation to a local ONS run by the GS1 Company Prefix Subscriber) under the "onsepc.com" namespace.

Let's suppose a user having the Canadian ONS Peer Root as the starting point for resolution would query for the same product using the following FQDN:

```
5.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.ons.epcglobalcanada.org
```

For the above query, the DNS resolver initially interrogates the Canadian ONS Peer Root. To redirect the query to "onsepc.com" (since the query is for a product under GS1 US namespace), the Canadian peer "epcglobalcanada.org" needs to know that the query is destined for the GS1 US ONS Peer Root. This is where DNAME delegation is required.

With the help of the GS1 prefix (last three digits "1.6.0" of the queried FQDN) "epcglobalcanada.org" can identify that the query is destined to the GS1 US ONS Peer Root.

To map a GS1 prefix to the corresponding ONS Peer Root and to append the appropriate peer domain name string to the query, there needs to be a mechanism within the existing DNS database.

The redirection SHALL be done based on the GS1 prefix using DNAME [RFC 2672] as follows:

```
1.6.0.gtin.gs1.id.ons.epcglobalcanada.org.          IN          DNAME
1.6.0.gtin.gs1.id.onsepc.com.
```

The facility with DNAME is that there is no necessity for each ONS Peer Root to have the knowledge about the zone information of its peers. The only necessity is that there should be DNAME redirection RRs for all GS1 prefixes for all MOs that participate in the ONS network. Each peer will have the DNAME resource records for all the GS1 MOs other than its own.

4.1.1. A Note about Alliance Numbers

Many MOs in their early years sublicensed GS1 GCPs from larger neighbours rather than managing their own. Typical of such MOs is GS1 Canada, formerly the Electronic Commerce Council of Canada, which sublicensed numerous GCPs from GS1 US, then the Uniform Code Council.

Reasons for such sublicensing are varied. In Canada's case, because the US is a significant trading partner and its retailers supported primarily the U.P.C. (now the GTIN-12) at the time, it was necessary to use US-assigned GCPs. Other countries may not have had MOs of their own at the time the GCPs were assigned by a neighbouring country but the management of those GCPs was delegated to the new country MO when the MO was chartered.

These numbers are referred to in GS1 as Alliance Numbers. They have the effect of fragmenting the GS1 namespace beyond the GS1 Prefix level. However, as there is no pattern to the Alliance Numbers, the delegation of queries for Alliance Numbers has to be the result of a bilateral agreement between the licensor MO and the licensee MO. When such delegation occurs, it SHALL be done using DNAME resource records.

4.1.2. Modification of the zone in the ONS Peer Root

There are multiple scenarios where a ONS Peer Root has to be updated with information pertaining to its peers. For example, suppose a GS1 MO decides not to manage its own ONS Peer Root but to be part of another existing peer. Similarly there may be a case where a small GS1 MO which has been part of a community ONS Peer Root decides to manage its own peer. There also cases wherein a new ONS Peer Root is added to the ONS network. In all these cases there needs to be a modification in the ONS Peer Root zone to correctly resolve to the modified peer.

The modification for the NS delegations at the concerned ONS Peer Root(s) has to be managed by the MO. For updating the DNAME redirections in the ONS Peer Roots a Peer Delegation file SHALL be used. This Peer Delegation file SHALL contain a list of valid ONS Peer Roots and the prefix ranges supported by each MO. This file SHALL be managed by GS1 Global Office and SHALL be accessible only to peers in the ONS network.

The Peer Delegation file is not used for resolution and in no way should the Peer Delegation file have a privileged position. The contents of the Peer Delegation file SHALL be used by each ONS Peer Root to update itself with the modifications that have occurred in any of its peers.

The Peer Delegation file SHALL be constructed according to the following XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:epcglobal:ons:peerDelegation:xsd:1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="urn:epcglobal:ons:peerDelegation:xsd:1">

    <xs:complexType name="PrefixRange">
        <xs:annotation>
            <xs:documentation>Defines a range of prefixes supported by the enclosing peer
node.</xs:documentation>
        </xs:annotation>
```



```
            <xs:sequence>
                <xs:element name="Low" minOccurs="1" maxOccurs="1">
                    <xs:annotation>
                        <xs:documentation>The low value (inclusive) of the prefix
range.</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:length value="3"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="High" minOccurs="1" maxOccurs="1">
                    <xs:annotation>
                        <xs:documentation>The high value (inclusive) of the prefix
range.</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:length value="3"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>

        <xs:complexType name="PeerNode">
            <xs:annotation>
                <xs:documentation>Defines the domain name and range of prefixes for a single node
in the ONS federation.</xs:documentation>
            </xs:annotation>
            <xs:sequence>
                <xs:element name="DomainName" type="xs:string" minOccurs="1" maxOccurs="1">
                    <xs:annotation>
                        <xs:documentation>The domain name of the peer node.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="PrefixRange" type="tns:PrefixRange" minOccurs="1"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>The prefix ranges supported by the peer
node.</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>

        <xs:complexType name="PeerDelegation">
            <xs:annotation>
                <xs:documentation>Defines the peer nodes involved in the ONS
delegation.</xs:documentation>
            </xs:annotation>
            <xs:sequence>
                <xs:element name="LastUpdated" type="xs:dateTime" minOccurs="1" maxOccurs="1">
                    <xs:annotation>
                        <xs:documentation>The date and time the file was last
updated.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="PeerNode" type="tns:PeerNode" minOccurs="1"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Peer nodes involved in the ONS
delegation.</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>

        <xs:element name="PeerDelegation" type="tns:PeerDelegation"/>
    </xs:schema>
```


In case there is a modification for a domain name for any GS1 Prefix, the responsible MO SHALL notify GS1 Global Office so that the Peer Delegation file may be updated. The Peer Delegation file is then updated appropriately and the “LastUpdated” field set to the current date and time.

Each peer accesses the Peer Delegation file at a scheduled time and compares the “LastUpdated” field with the date and time it was last used. In the event that the “LastUpdated” field is later than the last date and time the Peer Delegation file was used, the new Peer Delegation file is used to update the DNAME resource records for the ONS Peer Nodes. Each peer SHALL check for updates to the Peer Delegation file at most every 24 hours and SHALL update its local zone file if the file has been updated.

5. ONS Formal Specification

The formal specification of ONS is a set of technical rules and procedures to be followed by Publishers, Clients, and Peer Node Operators (MOs).

An ONS Client is an application that wishes to use ONS to identify a service that may provide information related to a specific GS1 Identification Key. An ONS Publisher is an entity responsible for making services available to ONS Clients by creating service pointer entries in an ONS Peer Root. An ONS Peer Root is an implementation of a DNS Server; because ONS differs from DNS only in what data is provided by the server, not in the operation of the server itself, there is no separate specification for an ONS Peer Root. Any DNS server compliant with [DNS] and [RFC 3403] may be used as an ONS Peer Root.

The ONS specification consists of four ingredients:

- The DDDS Application Specification (Section 6) for ONS.
- A procedure (Section 7.1) that an ONS Client SHALL follow in order to present a query to ONS. This procedure specifies how a GS1 Identification Key is converted to a DNS NAPTR query.
- A set of rules (Section 7.2) that ONS Publishers SHALL follow to represent ONS information (namely, pointers to services for GS1 Identification Keys) as DNS NAPTR records within an ONS Peer Root.
- A procedure (Section 8) that an ONS Client SHALL follow in order to interpret the results of an ONS query. This procedure specifies how an ONS Client can locate a service using the information provided by the ONS Peer Root.

6. ONS DDDS Application Specification

This section defines how the ONS client uses the methods explained in RFC 3402 and RFC 3403 to resolve a GS1 Identification Key. The DDDS database used by this application is the DNS database as explained in RFC 3403.

6.1. Application Unique String (AUS)

As per RFC 3402, the AUS is the initial input to a DDDS application. The DDDS application here is the ONS client.

The AUS is composed of:

- the lower case ISO 639-1 language code (optional) of the client querying the database;
- the lower case ISO 3166-1 alpha-2 country code (optional) of the client querying the database;

- the GS1 Identification Key type abbreviation in lower case; and
- the full-length GS1 Identification Key itself, less any serial number.

Each field in the AUS is separated by the vertical bar ('|', ASCII 124 decimal or 7C hexadecimal). This character has been chosen as it is not legitimately part of any of the fields within the AUS. Optional fields that have no value are encoded using empty strings; all field separators are required.

Valid GS1 Identification Key type abbreviations in lower case are:

Abbreviation (lower case)	GS1 Identification Key type
gtin	Global Trade Item Number
sscc	Serial Shipping Container Code
gln	Global Location Number
grai	Global Returnable Asset Identifier
giai	Global Individual Asset Identifier
gsrn	Global Service Relation Number
gdti	Global Document Type Identifier
gsin	Global Shipment Identification Number
ginc	Global Identification Number for Consignment

6.1.1. Converting an EPC to an AUS

In order to query the DNS for an EPC, the EPC URI form must be converted to a GS1 Identification Key. Note that this applies only to EPCs that are derived from GS1 Identification Keys; EPCs representing the General Identifier (GID), US Department of Defense Identifier (DOD), the Aerospace and Defense Identifier (ADI), and any others cannot be transformed into GS1 Identification Keys and are therefore not supported.

The procedure for the conversion is documented in Section 7 of the EPCglobal Tag Data Standards [EPC].

6.1.2. Examples (non-normative)

1. The GTIN-13 `0614141322602` for a client that is independent of language and country is represented as an AUS as:

   `||gtin|00614141322602`

   Note the addition of the digit '0' to the beginning to pad the GTIN to a full 14 digits.

2. The GIAI `0614141997uc-171+UX` for a client that is independent of language but is looking for a Brazilian service is represented as an AUS as:

   `|br|giai|0614141997uc-171+UX`

3. The GDTI `0614141665815` with serial number `999888777` for a client that is looking for an English service in Canada is represented as an AUS as:

   `en|ca|gdti|0614141665815`

4. The SSCC `306141417782246356` for a client that is looking for a French service in any country is represented as an AUS as:

   `fr||sscc|306141417782246356`

6.2. First Well-Known Rule

As per RFC 3402, the first well-known rule is a rewrite rule that is defined by the application (i.e. ONS client). This rule is used to produce the first valid key.

This rule applies only to GS1 Identification Keys; while ONS is not required to explicitly support non-GS1 identification systems, it must not explicitly impede such systems either. The first well-known rule is applied to the GS1 Identification Key and GS1 Identification Key type. In general, the format of the first valid key for ONS is:

```
<transformation of the identification key>.<identification key type>.<organisation namespace>.id.<valid ONS Peer Root domain name>
```

For GS1:

- transformation of the identification key is explained below;
- identification key type is the lower-case GS1 Identification Key type abbreviation defined in Section 6.1;
- organisation namespace is “gs1”; and
- valid ONS Peer Root domain name is one that is formally approved as a ONS Peer Root by GS1.

Note that non-GS1 identification systems which may implement ONS may use ONS domains wholly separate from those used by GS1.

The first valid key for a GS1 AUS as follows:

- Start with an empty string.
- Append a transformation of the GS1 Identification Key within the AUS.
  - Strip the optional serial number if applicable (GDTI and GRAI).
  - Strip the checksum digit if applicable (all except GIAI and GINC which, as alphanumeric GS1 Identification Keys, do not have a checksum digit).
  - Hold the leading digit (GTIN indicator digit, SSCC extension digit, GRAI zero) in its position if applicable.
  - Reverse all the remaining characters.
  - For each character:
    - If the character is a number or a lower-case letter, leave unmodified.
    - If the character is an upper-case letter, convert it to lower case and prefix it with the letter ‘u’.
    - Otherwise, convert the character to its two-digit hexadecimal equivalent and prefix it with the letter ‘x’.
  - For each character or its conversion as per the previous step, append a ‘.’ (period).
- Append the identification key type.
- Append “.gs1.id.”.
- Append a valid ONS Peer Root domain name.

Although a single AUS may be encoded as the first valid key in exactly the number of ways matching the number of ONS Peer Root domains (see the last step above), each instance shall be equivalent and shall resolve to the same results.

6.2.1. Examples (non-normative)

1. Given the GTIN-13 `0614141322602` as the primary identifier for a product, its representation as an AUS for a client that is independent of language and country is:

   `||gtin|00614141322602`

   Its representation as the first valid key with onsepc.com as the starting point for the query is:

   `0.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.onsepc.com`

   Its representation as the first valid key with ons.epcglobalcanada.org as the starting point for the query is:

   `0.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.ons.epcglobalcanada.org`

   Even though the string appended is different for the above two examples, they will finally resolve to the same service/information that is appropriate to the GS1 Identification Key. The two domains are equivalent.

2. Given the GTIN `50614141322607` as the primary identifier for a product, its representation as an AUS for a client that is independent of language but is looking for a Brazilian service is:

   `|br|gtin|50614141322607`

   Its representation as the first valid key with onsepc.com as the starting point for the query is:

   `5.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.onsepc.com`

   Its representation as the first valid key with ons.epcglobalcanada.org as the starting point for the query is:

   `5.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.ons.epcglobalcanada.org`

   The two domains are equivalent.

3. Given the `GIAI 0614141997uc-171+UX` as the primary identifier for an asset, its representation as an AUS for a client that is looking for an English service in Canada is:

   `en|ca|giai|0614141997uc-171+UX`

   This GS1 Identification Key has a mix of lower- and upper-case characters, digits, and punctuation characters. The lower-case characters and digits are translated as is, the upper-case characters are escaped with “u”, and the punctuation characters are escaped with “x” and their hexadecimal equivalent.

   Its representation as the first valid key with onsepc.com as the starting point for the query is:

   `ux.uu.x2b.1.7.1.x2d.c.u.7.9.9.1.4.1.4.1.6.0.giai.gs1.id.onsepc.com`

   Its representation as the first valid key with ons.epcglobalcanada.org as the starting point for the query is:

   `ux.uu.x2b.1.7.1.x2d.c.u.7.9.9.1.4.1.4.1.6.0.giai.gs1.id.ons.epcglobalcanada.org`

   The two domains are equivalent.

4. Given the GDTI `0614141665815` with serial number `999888777` as the primary identifier for a document, its representation as an AUS for a client that is looking for an English service in any country is:

   `en||gdti|0614141665815`

   Its representation as the first valid key with onsepc.com as the starting point for the query is:

   `1.8.5.6.6.1.4.1.4.1.6.0.gdti.gs1.id.onsepc.com`

Its representation as the first valid key with ons.epcglobalcanada.org as the starting point for the query is:

`1.8.5.6.6.1.4.1.4.1.6.0.gdti.gs1.id.ons.epcglobalcanada.org`

The two domains are equivalent.

5. Given the SSCC `306141417782246356` as the primary identifier for a shipping container, its representation as an AUS for a client that is looking for a French service in any country is:

   `fr||sscc|306141417782246356`

   Its representation as the first valid key with onsepc.com as the starting point for the query is:

   `3.5.3.6.4.2.2.8.7.7.1.4.1.4.1.6.0.sscc.gs1.id.onsepc.com`

   Its representation as the first valid key with ons.epcglobalcanada.org as the starting point for the query is:

   `3.5.3.6.4.2.2.8.7.7.1.4.1.4.1.6.0.sscc.gs1.id.ons.epcglobalcanada.org`

   The two domains are equivalent.

6.2.2. A note about non-numeric characters

The following statements are true:

1. The GCP is numeric only.
2. The first non-numeric character encountered in a GS1 Identification Key must therefore be beyond the GCP.
3. The GS1 Identification Keys that support non-numeric characters (GLN with extension, GRAI, GIAI, GDTI, and GINC) do so only at the instance or serial level, not at the class level.
4. “Section 4.3 – Granularity of identification” in the Federated ONS Requirements Document states that: “The granularity of identification (the lowest level at which queries can be made) shall be at minimum one level above the serial number in the identifier. ONS is about discovering class-level services only.”

It is therefore possible to stop encoding using the first well known rule the moment the first non-numeric character is encountered. However, there is no guarantee that the above restriction will be in place in the next iteration of ONS so to maximize the future benefits of ONS the domain name SHALL be encoded in its entirety as above.

7. ONS DDDS database

The DDDS database used for ONS resolution is the DNS. RFC 3403 specifies a DDDS Database that uses the NAPTR DNS resource record to contain the rewrite rules. The Keys for this database are encoded as domain-names. Section 6 explains how the keys are generated.

This domain-name is used to request NAPTR records which may contain the end result. A brief description of the contents of the database is explained in the following subsections.

7.1. DNS query format

The following specifies the procedure an ONS client SHALL follow to present a query to ONS for a GS1 Identification Key:

1. Begin with an AUS as defined in Section 6.1.

2. Follow the procedure in Section 6.2 to convert the AUS into a domain name, ending with the host domain name of a valid ONS Peer Root, for example:

   `5.0.6.2.2.3.1.4.1.4.1.6.0.gtin.id.gs1.onsepc.com`.

3. Use a DNS resolver to query for DNS Type Code 35 (NAPTR) records [RFC 3403] for the domain name from Step 2. The method for obtaining and using the DNS resolver is outside the scope of this specification. An ONS Client MAY use any DNS resolver conforming to [DNS], using whatever API is available.

7.2. NAPTR RR

ONS contains pointers to authoritative information for GS1 Identification Keys in the DNS database. Each such pointer takes the form of a DNS Type Code 35 (NAPTR) record [RFC 3403]. As explained in section 8.1, the ONS client resolves for the NAPTR records with the domain names as the input. This section specifies how ONS Publishers must encode information into NAPTR records.

The contents of a DNS NAPTR record are logically formatted as follows:

Order	Pref	Flags	Service	Regexp	Replacement
`0`	`0`	`u`	`(Service URL)`	`!^.*$!http://example.com/cgi-bin/epcis!`	`.` (a period)

ONS Publishers SHALL obey the following rules:

- The **Order** field SHALL be a non-negative integer.

  *Explanation (non-normative): The Order field is used in DNS applications where a series of regular expressions from distinct NAPTR records are applied consecutively to an input. Where multiple service implementations are available (e.g. segmented by language and/or country), the **Order** field specifies the order in which the rules are applied to the AUS.*

- The **Pref** field SHALL be a non-negative integer. The value of the **Pref** field is an ordinal that specifies that the service in one record is preferred to the service in another record having the same **Service** field. An ONS Client SHOULD attempt to use a service having a lower **Pref** number before using an equivalent service having a higher **Pref** number. The **Pref** field may be used, for example, to select a primary, high-capacity server over a secondary backup server to be used only when the primary server fails.

- The **Flags** field SHALL be set to 'u' to indicate that the **Regexp** field contains a URI OR 't' to indicate that the **Regexp** field contains plain text.

- The **Service** field contains an indicator of the type of service that can be found at the URI in question. This feature allows for the ONS service to indicate different services for different classes of service. The value of the Service field SHALL take the form of a URL and be interpreted as defined in Section 7.2.2.

- The **Regexp** field specifies an expression to be applied to the AUS together with a replacement for the AUS should there be a match to the expression. The value of this field SHALL be of the form !*expression*!*substitution*! where the exclamation mark is the field delimiter, *expression* is pattern to match to the AUS, and *substitution* is the replacement string, either a URI or text as appropriate (depending on the flag).

- The **Replacement** field is not used by ONS but as it is a special DNS field its value SHALL be set to a single period ('.') instead of simply a blank.

A detailed explanation of certain fields in the NAPTR resource records are given here:

7.2.1. Flags

The following flags are defined:

- "u" means that the output of the rule is a URI (typically a URL to a web service or a web page). Interpretation of the URI and the interaction model with the target (if any) of the URI is dependent on the service type identifier.
- "t" means that the output of the rule is plain text. Interpretation of the text is dependent on the service type identifier.

The remaining alphabetic flags are reserved for future versions of this specification. The numeric flags may be used for local experimentation. The flags are all mutually exclusive and resolution libraries MAY signal an error if more than one is given. (Experimental code and code for assisting in the creation of Rewrite Rules would be more likely to signal such an error than a client such as a browser.) Multiple flags may be allowed in the future, so implementers MUST NOT assume that the flags field can contain only 0 or 1 characters. Finally, if a client encounters a record with an unknown flag, it MUST ignore it and move to the next Rule. This test takes precedence over any ordering since flags can control the interpretation placed on fields. A novel flag might change the interpretation of the regex and/or replacement fields such that it is impossible to determine if a record matches a given target.

The above flags are terminal flags and they halt the looping of the DDDS algorithm. If none of the above flags is present, clients may assume that another Rule exists at the Key produced by the current Rewrite Rule.

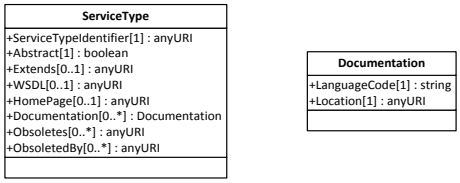
7.2.2. Services

ONS is required to support a dynamic service definition model wherein any service may be defined by GS1 Global, a GS1 Member Organization, an industry group, a member, a service provider, or in fact any entity that wishes to publish services based on GS1 identification keys in ONS. The service definitions need to support scope definitions at the global and local levels.

Service type identifiers for this Application SHALL take the form of a URL.

The service type URL on its own SHALL point to an XML document describing the service. Note that this does not imply that the XML document is publicly accessible; if the service type has restricted circulation (e.g. within an industry group), the organization defining the service type MAY require credentials to access the document (e.g. basic HTTP authentication, FTP username and password).

The XML document shall have the following structure:



The structure is represented by the following XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:epcglobal:ons:serviceType:xsd:1"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="urn:epcglobal:ons:serviceType:xsd:1">

    <xs:complexType name="ServiceType">
        <xs:sequence>
            <xs:element name="ServiceTypeIdentifier" type="xs:anyURI" minOccurs="1"
maxOccurs="1"/>
            <xs:element name="Abstract" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="Extends" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="TimeToLive" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="WSDL" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="HomePage" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
```



```
            <xs:element name="Documentation" type="tns:Documentation" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="Obsoletes" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="ObsoletedBy" type="xs:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Documentation">
        <xs:sequence>
            <xs:element name="LanguageCode" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="Location" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="ServiceType" type="tns:ServiceType"/>

</xs:schema>
```

7.2.2.1. Service Type Class

The ServiceType class is the definition of the service type itself.

Field	Description
ServiceTypeIdentifier	The service type identifier, also the URL at which the XML representation of this structure may be found.
Abstract	True if this is an abstract service type. An abstract service type is one that is used as the base for other service types rather than having a concrete implementation of its own. An example of an abstract service type is a generic data retrieval service that returns structured data whose contents are specified more concretely by the service types that extend this service type.
Extends	The service type identifier that this service type extends if any.
TimeToLive	The time to live (TTL) in minutes. The document may be cached by a client only for as long as specified in this field. This allows a publisher to be certain that updates to fields subsequent to this one will be picked up in a timely fashion. The recommended value is 10080 minutes (one week).
WSDL	The location of the Web Service Description Language file if any. Not all service types will be based on Web Services but for those that do, this field is required.
HomePage	The location of a human-readable home page if any. Typically, the home page for a service type is a page that provides a basic outline for the service type and may contain links to other, related service types. The presence, structure, and content of a home page are entirely at the discretion of the service type author.
Documentation	Documentation, possibly in multiple languages, for this service type. Supplements the documentation of the "Extends" service type if any.

Field	Description
Obsoletes	A list of service type identifiers that this service type obsoletes. As services evolve, new versions will occasionally replace old versions; this provides the necessary continuity between versions. The target service types should have the service type identifier for this service type in their "ObsoletedBy" list.
ObsoletedBy	A list of service type identifiers that obsolete this service type. As services evolve, new versions will occasionally replace old versions; this provides the necessary continuity between versions. The target service types should have the service type identifier for this service type in their "Obsoletes" list.

7.2.2.2. Documentation class

The service type documentation in a given language. This is a contained class that exists only within the context of the service type. Documentation may or may not exist for a service type at the discretion of the service type definer.

Field	Description
LanguageCode	The ISO language code representing the language in which the human-readable documentation is written, unique within the service type.
Location	The URL for human-readable documentation in the language specified. The presence of the URL field does not mean that the target is publicly accessible.

7.2.3. Regular expressions (non-normative)

The reason the **Regexp** field is in the form of a regular expression is that the NAPTR record is used by other applications that have the need to conditionally rewrite the URI or text to include other information. While none of the examples here make use of this feature, it has not been determined if this will always be the case. In the future it may become necessary to allow full regular expression and replacement functions within the Regexp field. Therefore, implementers would be wise to not assume that the URI or text can simply be extracted without any regular expression processing.

The general form of the Regexp field is as a Posix Extended Regular Expression. This form states that the first character encountered is the field delimiter between the regular expression and the replacement portion of the entire rewrite expression. In the example at the beginning of this Section 7.2 the delimiter is the exclamation point (!) character. The regular expression portion in most cases is ^.*$ which equates to 'match anything'. The substitution portion is text (for text services) or a URI (e.g. `http://example.com/cgi-bin/epcis`). The choice of '!' as the delimiter instead of a more traditional '/' makes the entire line much easier to read and less error prone.

7.2.4. Service type hierarchy (non-normative)

An application may be interested in multiple services, either directly by supporting multiple interaction models with multiple service types, indirectly by supporting a service type and all service types derived from it (see Section 7.2.2), or a combination of the two. Because of this, the necessary steps after issuing the query are to filter the results by service type and then group by service type.

The concept of inheritance in services is no different from that as applied to traditional object-oriented programming languages. Take, for example, the concept of a Product Recall in a consumer-based Mobile Commerce application.

In general, a Product Recall describes what is being recalled (e.g. product and set of lot numbers), how to identify what is being recalled (e.g. locating the lot number on the package), and what to do to process the recall (e.g. returning to the store from which it was purchased).

Local regulations may require additional data, such as a consumer hotline number to call for more information. As a result, there may be not only a global Product Recall service, but one defined for Canada, another for Germany, and so on. The country-specific Product Recall service types each have the URL of the global Product Recall service type in their “Extends” field.

A Canadian consumer, Alice, has a Mobile Commerce application, customized for Canada, on her device. In addition to displaying product data, images, and consumer reviews, the application also has the ability to discover and interact with a Product Recall service. She is in Canada and scans a product; through the ONS lookup process, she finds the Canadian Product Recall service, calls it, and is given either a null response (product is not subject to a recall) or full details of the recall, including any Canadian extensions.

Another Canadian consumer, Bob, has a similar application, but one that does not understand the Canadian extensions. He scans the same product, looking for the global Product Recall service type. The application doesn’t find it, but it does find the Canadian one, and inspecting the service type reveals that the Canadian service is just an extension of the global service. Bob’s application can interact with the Canadian, but only with the methods and data structures supported by the global service. Nevertheless, Bob gets the data he is looking for.

Alice now travels the Germany. A product scanned in Germany is likely to return the German Product Recall service type, not the global or the Canadian one. In order for Alice’s application to work, it has to understand not only the specific Canadian Product Recall service type but also the global Product Recall service type so that it can interact with the German Product Recall services in the same manner as Bob’s application.

For further details and a flow diagram for discovering derived service types, see Section 9.4.

8. Processing ONS Query Responses

ONS Clients SHALL use the following procedure to interpret the results returned by an ONS query as formulated in Section 7.1.

1. The result from the ONS query is a set of NAPTR records as described in Section 7.2. If no entry exists for the query for whatever reason, the result SHALL be considered to be an empty set.
2. Remove all records from Step 1 where the Service field is not supported by the application.
   a. A Service is supported by the application if it is explicitly declared to be supported or one of its base classes is declared to be supported and extensions of that base class are permitted by the application.
3. Group the results from Step 2 by the Service field.
4. For each group in the results from Step 3:
   a. Order the results according to the Order field.
   b. For each result, apply the Regexp substitution expression to the AUS. If a match is found, continue only with those results where the Order field is the same as the one in which there is the first match.
   c. From among the results in Step b, select those with the lowest Pref field value.
   d. From among the results in Step c, select a record at random.

e. Interpret the result in Step d as determined by the Flags.

   i. For a text field, apply the text as per the rules of the Service.

   ii. For a URI field, use the URI as per the rules of the Service.

f. If Step e is not successful, repeat Step d, excluding this and all previously failed selections. If all records from Step c have been tried, from among the results in Step b, select those having the next lowest value in the Pref field and return to Step d. If all records from Step b have been tried, stop: no service is available.

9. Examples (non-normative)

In the following examples the GS1 Identification Key in question is the GTIN `50614141322607` which represents a case of Example Corporation Model 100 Widgets.

The ONS client attempts to learn about this product by first following the procedure in Section 4, which converts the GS1 Identification Key into this domain name:

```
5.0.6.2.2.3.1.4.1.4.1.6.0.gtin.gs1.id.onsepc.com
```

The application then queries the DNS for NAPTR records for that domain name and receives the following records (The record number column is for reference only and not part of the result set):

#	Order	Pref	Flags	Service	Regexp	Replacement
1	0	0	t	http://www.gs1.org/ons/prefix	!^.*$!06,7!	.
2	0	0	u	http://www.gs1.org/ons/epcis	!^.*$!http://epcis.example.com/!	.
3	0	0	u	http://www.gs1.org/ons/mobilecomm	!^fr|ca|.*$!http://fr-ca.example.com/!	.
4	1	0	u	http://www.gs1.org/ons/mobilecomm	!^[a-z]*|fr|.*$!http://all-fr.example.com/!	.
5	2	0	u	http://www.gs1.org/ons/mobilecomm	!^fr|.*$!http://fr-all.example.com/!	.
6	3	0	u	http://www.gs1.org/ons/mobilecomm	!^es|us|.*$!http://es-us1.example.com/!	.
7	3	1	u	http://www.gs1.org/ons/mobilecomm	!^es|us|.*$!http://es-us2.example.com/!	.
8	4	0	u	http://www.gs1.org/ons/mobilecomm	!^.*$!http://en-all.example.com/!	.

778 Note: All of the URLs in this table are fictitious and do not represent any standard current or proposed. Final URLs for services defined by GS1
779 may not match those shown here.

Each of these records conforms to the rules specified in Section 7. The interpretation is as follows (the numbers in the list below match the record number on the previous page):

1. In this example, the service type represents a parsing of the GS1 Company Prefix. The flags specify text as the record type, which in this case represents the leading digits of the GS1 Company Prefix and its length. The string “06,7” means that all GS1 Company Prefixes starting with “06” are seven digits long; the GS1 Company Prefix is therefore “0614141”.
2. The service type represents EPCIS. The flags specify a URI as the record type: the regular expression substitution results in the URL of the EPCIS server for every AUS (“^.*$” matches all strings).
3. The service type represents a Mobile Commerce data provider. The flags specify a URI as the record type: the regular expression substitution results in the URL of the Mobile Commerce server for French Canadian data. The order of “0” (the lowest in all equal service types) means that this is the first record whose regular expression is applied to the client AUS.
4. Same service type as before, but is applied to the client AUS if the previous regular expression produced no match. The regular expression matches all clients operating in France (note the character set match for all languages).
5. Same service type as before, but is applied to the client AUS if the previous regular expression produced no match. The regular expression matches all clients operating in French in all countries.
6. Same service type as before, but is applied to the client AUS if the previous regular expression produced no match. The regular expression matches all clients operating in Spanish in the United States.
7. This is the same as the previous (the order is the same) and so will match the string as well, but will return a different URL as a result of the substitution and has a different preference. The URL in this record will be used if the first URL does not respond.
8. Same service type as before, but is applied to the client AUS if the previous regular expression produced no match. The regular expression matches all clients operating in all countries.

Finally, depending on the service that the ONS Client desires, it uses one or more of the records returned to locate an appropriate service. The following sections describe specific examples of services an ONS Client might locate. In each case, the ONS Client uses the procedure specified in Section 8 to locate a service, using the records returned above.

9.1. Finding the length of the GS1 Company Prefix (GCP)

One of the simplest but most powerful examples is association some application-specific text with a GS1 Identification Key. In the above example, the Service URL `http://www.gs1.org/ons/prefix` is associated with the length of the GCP. The text returned is interpreted as “for any GS1 Identification Key starting with ‘06’, the length of the GCP is 7”. This allows applications that need to encode RFID tags for a product where the length of the GCP is not previously known (e.g. when a distributor has to add RFID tags to a manufacturer’s product) to encode the tags properly.

The application issues the query and receives the NAPTR records above. It iterates through the list looking for the required Service URL which returns text that can be parsed by an appropriate application to determine the length of the GCP. It locates that service in the first record and returns the text found in the Regexp field. The application then parses the text according to the rules defined by the service.

9.2. Finding an authoritative EPCIS server for a product

This example shows how a GS1 Identification Key can be used to retrieve a pointer to an EPCIS server associated with the owner of the GS1 Identification Key. Again, using the same results from above, the client uses the second record and extracts the URI from the Regexp. It then uses that URI as the end point to which to send the EPCIS query.

9.3. Finding a Mobile Commerce service for product

This example shows how to interpret multiple instances of Mobile Commerce services according to the language and country requirements of a client.

A client looking for information in French in Canada would choose record 3. A client looking for information in any language in France would choose record 4. A client looking for information in French in any country would choose record 5.

A client looking for information in Spanish in the United States would choose records 6 and 7; the client would first try to access the URL in record 6 and only if that fails would it fall over to the URL in record 7.

All other clients would choose record 8 as the default as it matches all languages and countries.

9.4. Dynamic interaction with extended services

The ability to define one service as an extension of another is particularly useful when dealing with web services that have to interact in certain ways, typically to retrieve data.

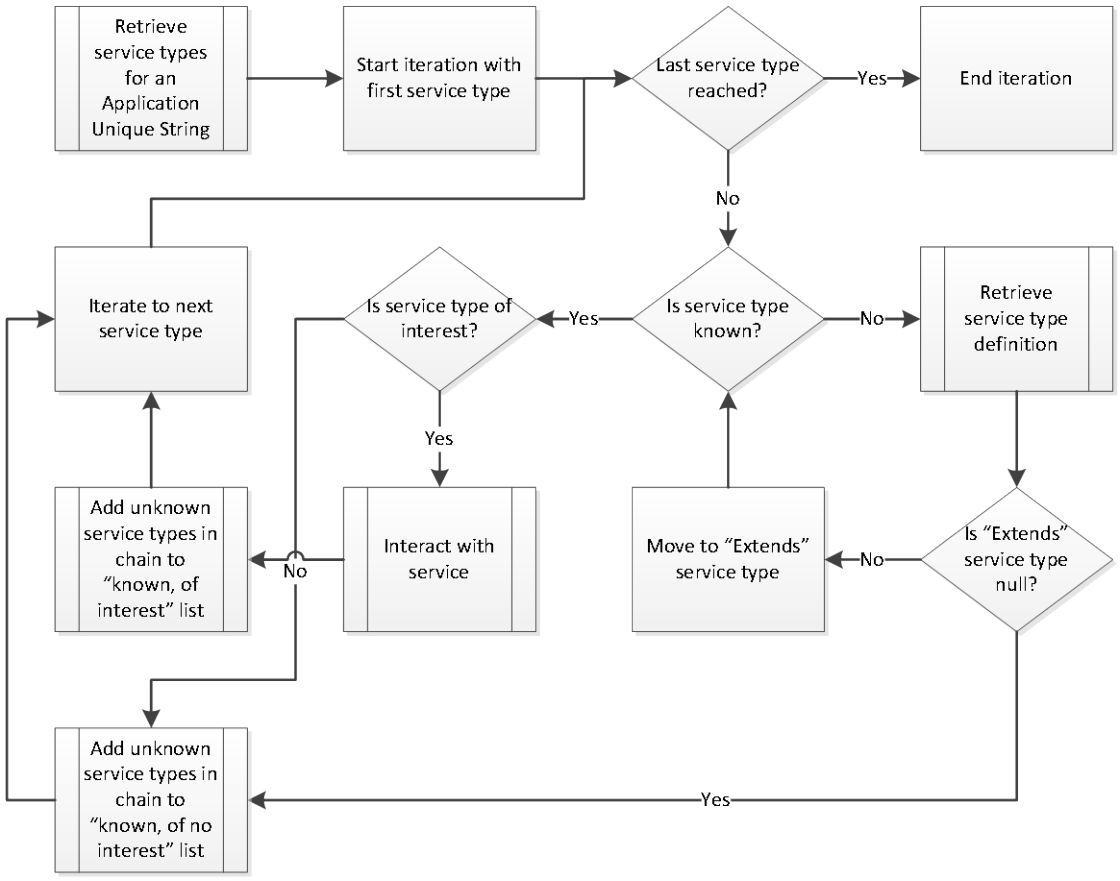
Assume, for example, that GS1 has defined an abstract service type with a single “getData()” method that takes a GS1 Identification Key as a parameter. The return type for this method is a basic object type with no attributes of interest.

Now assume that the Mobile Commerce working group has defined a number of data objects for products: nutrition data, allergen data, health certification data, and so on. Each of these data objects has a corresponding service type to define their particular implementation of the “getData()” method.

A Mobile Commerce application that is aware of these service types can easily discover them when querying ONS and can interact with services that are identified by these service types. When dealing with nutrition data, for example, the Mobile Commerce application can trigger warnings when the nutrition content of the product exceeds the user’s dietary requirements.

Suppose, however, that the Mobile Commerce developer wants to design the application to retrieve any data defined for a product, not just those that are hard-coded into the application. In theory, the application can interact with any service type that extends the generic data retrieval service type; it can’t interpret the data as it has no logic to understand its meaning, but it can at least display it to the user.

The discovery and interaction model would look like something like this:



Using this model, the Mobile Commerce application can interact with any service type that implements the generic data retrieval framework. For performance purposes, it can build up its list of known service types so that it can limit the number of queries to retrieve the service type definition.

9.5. Sample ServiceType XML

The following is a sample ServiceType XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:ServiceType xmlns:tns="urn:epcglobal:ons:serviceType:xsd:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.gs1.org/ons/xsd/1/ServiceType.xsd">

<ServiceTypeIdentifier>http://www.example.com/ons/SampleServiceType.xml</ServiceTypeIdentifier
>
  <Abstract>false</Abstract>
  <Extends>http://www.example.com/ons/SampleBaseServiceType.xml</Extends>
  <TimeToLive>10080</TimeToLive>
  <WSDL>http://service.example.com/services/SampleService?wsdl</WSDL>
  <HomePage>http://www.example.com/projects/SampleProject.xhtml</HomePage>
  <Documentation>
    <LanguageCode>en</LanguageCode>
    <Location>http://www.example.com/projects/en/SampleServiceType.xhtml</Location>
  </Documentation>
  <Documentation>
    <LanguageCode>fr</LanguageCode>
    <Location>http://www.example.com/projects/fr/SampleServiceType.xhtml</Location>
  </Documentation>
</tns:ServiceType>
```

The service type XML is self-referential; its location on the Internet is `http://www.example.com/ons/SampleServiceType.xml`. It is a concrete (not abstract) service type. It

extends the base service type `http://service.example.com/services/SampleService?wsdl`, has a WSDL at `http://service.example.com/services/SampleService?wsdl`, and has a human-readable home page at `http://www.example.com/projects/SampleProject.xhtml`. Detailed documentation is available in English and in French.

It should be pointed out that the suggested nomenclature here does not preclude that the same home page being shared by multiple service types. The example here shows that the home page is based on a project, which may define multiple service types. The same applies to documentation; there is no reason for a documentation page to apply to a single service type, though the names of the documentation URLs suggest that they apply to this service type only.

10. References

EPC

EPCglobal, "EPCglobal Tag Data Standards Version 1.6," EPCglobal Ratified Standard, September 2011. (See http://www.epcglobalinc.org/standards/tds)

DNS

(Internet Engineering Task Force). STD0013, *RFC 1034, RFC 1035*, ed Mockapetris, P. 2000. (See http://www.ietf.org/rfc/std/std13.txt)

RFC 2396

T. Berners-Lee, R. Fielding, L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax*, 1998. (See http://www.ietf.org/rfc/rfc2396.txt)

RFC 2672

Crawford, P. *Non-Terminal DNS Name Redirection*, 1999. (See http://www.ietf.org/rfc/rfc2672.txt)

RFC 3403

Mealling, Michael. *Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database*, 2002. (See http://www.ietf.org/rfc/rfc3403.txt)

11. Appendix A – Glossary (non-normative)

Auto-ID

"Automatic Identification" – a set of technologies used to identify anything, anywhere, automatically.

Berkeley Internet Name Domain

The Berkeley Internet Name Domain is the most widely used DNS implementation on the Internet.

BIND

See Berkeley Internet Name Domain.

Domain name

A hierarchical, 'dot' (.) separated namespace used to identify hosts on the Internet

DNS

See Domain Name System

Domain Name System

An infrastructure-level Internet service used to discover information about a domain name. It was originally developed to map a host name to an IP address, but has since been extended to other uses (such as ENUM, which maps a phone number to one or more communication services.

EPCIS

EPC Information Services – A GS1 EPCglobal specification that defines capture and query interfaces for event and other data.

FQDN

A fully qualified domain name (FQDN), sometimes also referred as an absolute domain name, is a domain name that specifies its exact location in the tree hierarchy of the DNS. It specifies all domain levels, including the top-level domain and the root domain. A fully qualified domain name is distinguished by its unambiguity; it can only be interpreted one way.

Member Organization

A Member Organization is a GS1 representative office with local authority over the management of the GS1 identification space and the promotion of its standards within a specific geographic area, typically a single country.

MO

See Member Organization.

NAPTR

"Naming Authority PoinTeR" -- A DNS record type (35) that contains information about a specific delegation point within some other namespace using regular expressions.

Object Name Service

A resolution system, based on DNS, for discovering authoritative data and services related to a GS1 Identification Key.

ONS

See Object Name Service.

Object Class code

A code that identifies a particular type of object that is created by a particular manufacturer

ONS Peer Root

A node within the ONS hierarchy that may be used as the starting point for an ONS query. Every ONS Peer Root is equivalent to every other in that a query that starts at ONS Peer Root A resolves to the same results as the same query that starts at ONS Peer Root B.

Reader

A radio enabled device that communicates with a tag.

RFID

"Radio Frequency Identification" -- A method of identifying unique items using radio waves. The big advantage over bar code technology is lasers must see a bar code to read it. Radio waves do not require line of sight and can pass through materials such as cardboard and plastic.

Regular Expression

A standard language for pattern matching within a string of characters and for composing new strings based on matched subcomponents of the original string (i.e. a search and replace function)

Serial Number

A number that identifies a particular instance of an object class.

tag

A microchip and antenna combo that is attached to a product. When activated by a tag Reader the tag emits its EPC plus other data it may have

URI

"Uniform Resource Identifier" -- the superclass of all identifiers that follow the 'scheme:scheme-specific-string' convention as specified in RFC 2396 [RFC2396] (e.g., "urn:isbn:2-9700369-0-8" or "http://example.com/news.html")

URL

"Uniform Resource Locator" -- a URI that identifies a resource via a representation of its primary access mechanism (e.g., its network “location”), rather than identifying the resource by name or by some other attribute(s) of that resource.

