



1
2 EPCglobal Object Name Service (ONS) 1.0.1

3 **Ratified Standard Specification with Approved, Fixed Errata**

4 **May 29, 2008**

5
6 This version: 1.0.1

7 Previous version: 1.0

8
9 **Disclaimer**

10 EPCglobal Inc™ is providing this document as a service to interested industries. This
11 document was developed through a consensus process of interested parties.
12 Although efforts have been to assure that the document is correct, reliable, and
13 technically accurate, EPCglobal Inc makes NO WARRANTY, EXPRESS OR IMPLIED,
14 THAT THIS DOCUMENT IS CORRECT, WILL NOT REQUIRE
15 MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE,
16 OR WILL BE SUITABLE FOR ANY PURPOSE OR WORKABLE IN ANY
17 APPLICATION, OR OTHERWISE. Use of this document is with the
18 understanding that EPCglobal Inc has no liability for any claim to the contrary, or for
19 any damage or loss of any kind or nature.

20 **Copyright notice**

21 © 2008, EPCglobal Inc.

22 All rights reserved. Unauthorized reproduction, modification, and/or use of this document is not
23 permitted. Requests for permission to reproduce should be addressed to
24 epcglobal@epcglobalinc.org.

25
26 EPCglobal Inc.™ is providing this document as a service to interested industries. This document was
27 developed through a consensus process of interested parties. Although efforts have been to assure
28 that the document is correct, reliable, and technically accurate, EPCglobal Inc. makes NO
29 WARRANTY, EXPRESS OR IMPLIED, THAT THIS DOCUMENT IS CORRECT, WILL NOT
30 REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGICAL ADVANCES DICTATE, OR
31 WILL BE SUITABLE FOR ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR
32 OTHERWISE. Use of this Document is with the understanding that EPCglobal Inc. has no liability for
33 any claim to the contrary, or for any damage or loss of any kind or nature

34 Abstract

35 This document specifies how the Domain Name System is used to locate authoritative metadata
36 and services associated with a given Electronic Product Code (EPC). Its target audience is
37 developers that will be implementing ONS resolution systems for applications.

38 Status of this document

39 This section describes the status of this document at the time of its publication. Other
40 documents may supersede this document. The latest status of this document series is maintained
41 at EPCglobal. This document was ratified by the EPCglobal Board of Governors on October 4,
42 2005. This version, 1.0.1 corrects errata found in the originally, ratified version of ONS 1.0.

43 Comments on this document should be sent to the EPCglobal Software Action Group mailing
44 list (sag@lists.epcglobalinc.org).

45 Fixed Errata

Section#	Line #	Description	Disposition
Cover Page		Cover Page does not match other EPCglobal Standards	Added Disclaimers, Copyright notice, revision date and GS1/EPCglobal Logo.
Status		Update status box	List nature of changes to document included
1	70-75	Remove SGTIN and last sentence in the paragraph	
4.1	124	Deleted [Tag Data Standard]	
4.1	131-135	Delete last two sentences of 3d paragraph	
4.1	142	Deleted [RFC 2826]	
4.2	148-149	Deleted last sentence	
4.2	152-154	Deleted sentence that begins with "The yellow components..."	
5.1.1	249-253	Delete sentences in first paragraph beginning with "Even the concept..." and "Since DNS conveys..." Also deleted the word "Therefore at the beginning of the next sentence.	
5.1.1	263-264	Deleted new string[] } from box	
5.1.1	268	Delete the word "secondaries	Replaced with secondary servers
5.2	300 & 319	Deleted [Tag Data Standard]	
5.2.1	340-341	Deleted the words "what is normally considered the"	
6.1	356-359	Deleted last two sentences in the first paragraph	Added: For the sake of discussion below, the SGTIN form of EPC is used as an example. Similar principles apply to the other forms of

			EPC.
6.1	377-378	Delete two occurrences of IP addresses	Replace with host names
6.1	381-383	Delete this paragraph/sentence	
7	407,409,412	Delete MUST	Replace it with SHALL
8	416	Delete MUST	Replace it with SHALL
8	419-420,438-439, 444	Deleted [Tag Data Standard]	
9	460, 461, 467,472,475,484	Delete MUST	Replace it with SHALL
10	509	Delete MUST	Replace it with SHALL
12	583-622	References Updated	
13	662-664	Delete PML definition	
13	682-684	Deleted definition starting at “the superclass of all identifiers... html”)	Replaced with “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.”

46

47 **Table of contents**

48 1 Introduction 6

49 2 Terminology and Typographical Conventions..... 6

50 3 Background Information (non-normative)..... 6

51 4 EPC System Network Architecture (non-normative)..... 6

52 4.1 Electronic Product Code (EPC)..... 7

53 4.2 EPC Network Software Architecture Components 7

54 5 ONS Introduction 10

55 5.1 The Domain Name System (DNS) (non-normative)..... 10

56 5.1.1 Client’s View 11

57 5.1.2 Publisher’s View 11

58 5.2 ONS’s Usage of DNS 12

59 5.2.1 Serial Number Level Queries to the ONS..... 14

60 6 ONS Nameserver Infrastructure Organization (non-normative)..... 14

61 6.1 ONS Delegation Rules..... 14

62 6.2 Zone Maintenance Guidelines 15

63 7 ONS Formal Specification 15

64 8 DNS Query Format 16

65 9 DNS Records for ONS 17

66 10 Processing ONS Query Responses 18

67 11 Examples (non-normative)..... 18

68 11.1 Finding a WSDL file for a product 19

69 11.2 Finding an authoritative EPCIS server for a product 19

70 11.3 Finding an HTML formatted web page description of a product 19

71 11.4 Finding an XML-RPC gateway to the Web Service interfaces 20

72 12 References..... 20

73 13 Appendix A – Glossary (non-normative) 21

74 14 Appendix B -- DDDS Application Specification (non-normative) 22

75 14.1 Application Unique String 23

76 14.2 First Well Known Rule 23

77 14.3 Expected Output..... 23

78 14.4 Valid Databases..... 23

79	14.5	Valid Flags	23
80	14.6	Service Parameters	23
81	15	Appendix C -- Service Field Registrations (non-normative).....	24
82	15.1	Registration Template	24
83	15.2	Service Registrations.....	24
84			
85			

86 **1 Introduction**

87 This document specifies how the Domain Name System is used to locate authoritative metadata
88 and services associated with a given Electronic Product Code (EPC). Its target audience is
89 developers that will be implementing ONS resolution systems for applications.

90 **2 Terminology and Typographical Conventions**

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

96 All sections of this document are normative, except where explicitly noted as non-normative.

97 The following typographical conventions are used throughout the document:

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

104 **3 Background Information (non-normative)**

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

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

115 **4 EPC System Network Architecture (non-normative)**

116 Radio Frequency Identification is a technology used to identify, track and locate assets. The
117 vision that drives the developments of EPCglobal is the universal unique identification of
118 individual items. The unique number, called an EPC (Electronic Product Code) will be encoded
119 in an inexpensive Radio Frequency Identification (RFID) tag. The EPC Network will also
120 capture and make available (via the Internet and for authorized requests) other information that
121 pertains to a given item to authorized requestors.

122 **4.1 Electronic Product Code (EPC)**

123 The EPC Network architecture provides a method for the inclusion of commercial (both
124 physical and otherwise) products within a network of information services. This architecture
125 makes several axiomatic assumptions, the most important being that it should leverage existing
126 Internet technology and infrastructure as much as possible. As such, it adheres to the "hour
127 glass model" [Willinger and Doyle] of the Internet by standardizing on one identifier scheme:
128 the Electronic Product Code (EPC) [EPC].

129 In most situations the EPC will denote some physical object. EPC identifiers are divided into
130 groups, or *namespaces*. Each of these namespaces corresponds to a particular subset of items
131 that can be identified. For example, XML Schemas are denoted using the 'xml' namespace, raw
132 RFID tag contents are kept in the 'raw' namespace. The 'id' namespace is generally reserved
133 for EPCs that can be encoded onto RFID tags and for which services may be looked up using
134 ONS. This 'id' namespace is further subdivided into sub-namespaces corresponding to different
135 naming schemes for physical objects, including Serialized GTINs, SSCCs, GLNs, etc. These
136 namespaces are defined normatively in the EPCglobal Tag Data Standards [EPC].

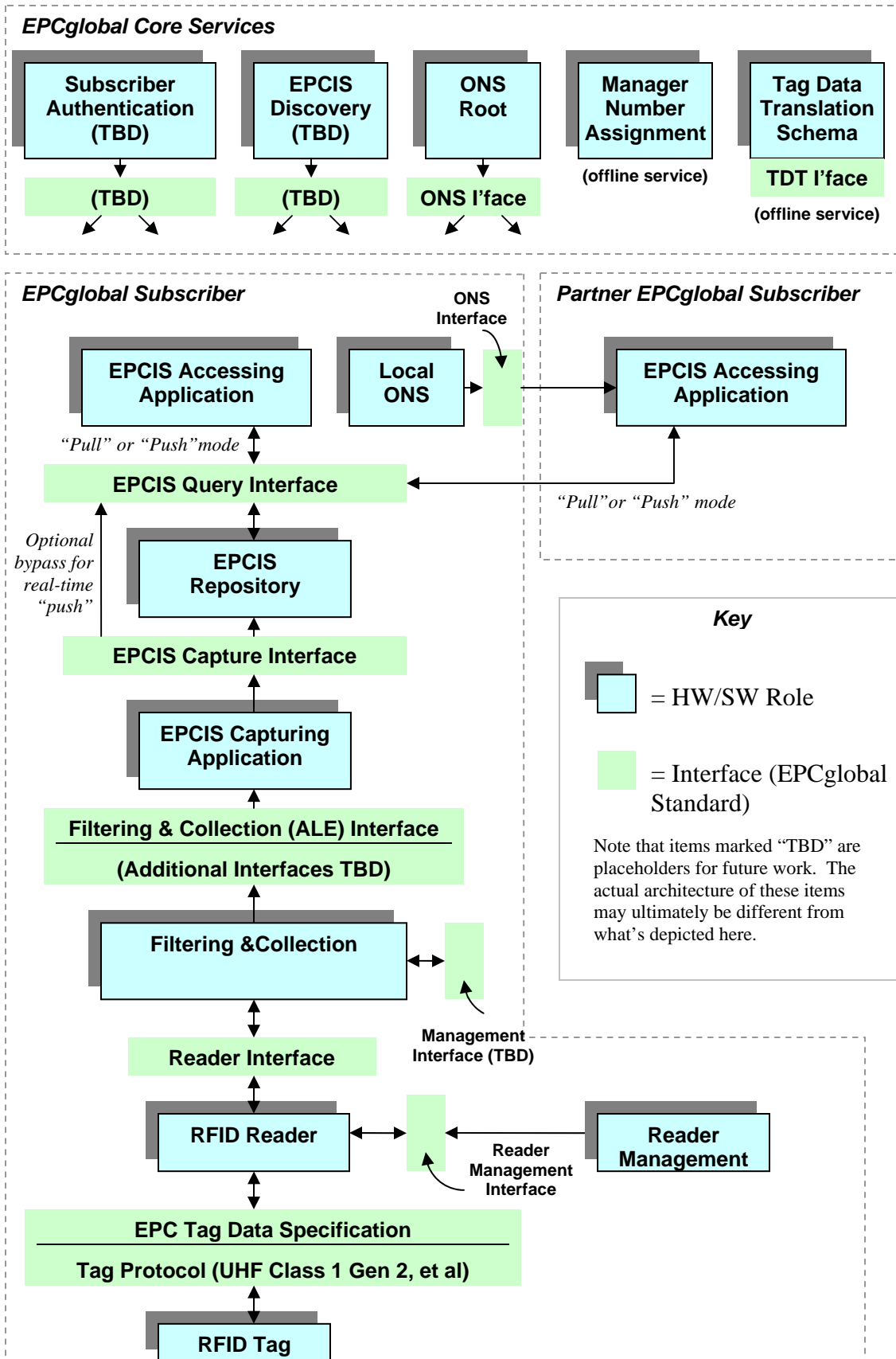
137 Each of the sub-namespaces that are defined by the Tag Data Standards specification have a
138 slightly different structure depending on what they identify, how they are used, and how they
139 are assigned. The SGTIN is used to identify an individual product that is assigned by the
140 company that creates that product. Thus the SGTIN contains a Manager Number, an Object
141 Class, and a Serial Number. Other sub-namespaces such as the SSCC go directly from the
142 Manager Number to the Serial Number and have no concept of an "Object Class".

143 In order to further leverage the use of Internet derived technology and systems, the EPC is
144 encoded as a Uniform Resource Identifier (URI). URIs are the basic addressing scheme for the
145 entire World Wide Web and ensure that the EPC Network is compatible with the Internet going
146 forward.

147 While an addressing scheme by itself is useful, it can only be used within a network when a
148 mechanism is provided to **authoritatively** look up information about that identifier. This EPC
149 'resolution' mechanism is called the Object Naming Service, or ONS and is what forms the
150 core integrating, or 'truth' verifying, principle of the EPC Network.

151 **4.2 EPC Network Software Architecture Components**

152 The EPC Network Architecture as in Fig. 1 shows the high-level components of the EPC
153 Network. This section highlights and makes reference to The EPCglobal Architecture
154 Framework Version 1, July 2005 [EPCAF].



155
156

Figure 1: EPC Network Architecture: Components and Layers

157 Components in blue are “roles”, not discrete pieces of software. Components in green are
158 interfaces used between two roles. These components from the figures above are described in
159 the sections below:

- 160 • *Readers* that make multiple observations of RFID tags while they are in the read zone.
- 161 • *Reader Protocol Interface* Delivers raw tag reads from Readers to Middleware. Events at
162 this interface say “Reader A saw EPC X at time T.”
- 163 • *Middleware* Accumulates and filters raw tag reads
- 164 • *ALE Interface* Delivers consolidated, filtered tag read data from Middleware to Local
165 Application. Events at this interface are “At Location L, between time T1 and T2, the
166 following EPCs were observed” where the list of EPCs has no duplicates and has been
167 filtered by appropriate criteria.
- 168 • *EPC Capturing Application* Recognizes the occurrence of EPC-related business events,
169 and delivers these as EPCIS data
- 170 • *EPCIS Capture Interface* Provides a path for communicating EPCIS events generated by
171 EPCIS Capturing Applications to other roles that require them, including EPCIS
172 Repositories, internal EPCIS Accessing Applications, and Partner EPCIS Accessing
173 Applications.
- 174 • *EPCIS Repository* Records EPCIS-level events generated by one or more EPCIS Capturing
175 Applications, and makes them available for later query by EPCIS Accessing Applications.
- 176 • *EPCIS Query Interface* Provides means whereby an EPCIS Accessing Application can
177 request EPCIS data from an EPCIS Repository or an EPCIS Capturing Application, and the
178 means by which the result is returned.
- 179 • *EPCIS-Accessing Application* Software that carries out overall enterprise business
180 processes, such as warehouse management, shipping and receiving, historical throughput
181 analysis, and so forth, aided by EPC-related data.
- 182 • *Local ONS* Fulfills ONS lookup requests for EPCs within the control of the enterprise that
183 operates the Local ONS; that is, EPCs for which the enterprise is the EPC Manager.
- 184 • *EPCIS Accessing Application* An EPCIS-enabled Application of a trading partner.
185 Partner Applications may be granted access to a subset of the information that is available
186 within the enterprise.
- 187 • *Tag Data Translation Schema* Provides a machine-readable file that defines how to
188 translate between EPC encodings defined by the EPC Tag Data Specification. EPCglobal
189 provides this file for use by End-users, so that components of their infrastructure may
190 automatically become aware of new EPC formats as they are defined.
- 191 • *Manager Number Assignment* Ensures global uniqueness of EPCs by maintaining
192 uniqueness of EPC Manager Numbers assigned to EPCglobal Subscribers
- 193 • *Object Name Service (ONS) Root* A service that, given an EPC, can return a list of
194 network accessible service endpoints that pertain to the EPC in question. ONS **does not**
195 contain actual data about the EPC. It only contains the network address of services that
196 contain the actual data. ONS is also **authoritative** in that the entity that has change control
197 over the information about the EPC is the same entity that assigned the EPC to the item to

198 begin with. For example, in the case of an SGTIN EPC, the entity having control over the
199 ONS record is the owner of the SGTIN manager number (EAN.UCC Company Prefix).

200 • *EPC Discovery Service(s)* A “search engine” for EPC related data. A Discovery Service
201 returns locations that have some data related to an EPC. Unlike ONS, in general a
202 Discovery Service may contain pointers to entities other than the entity that originally
203 assigned the EPC code. Hence, Discovery Services are **not universally authoritative** for
204 any data they may have about an EPC. It is expected that there will be multiple
205 competitively-run Discovery Services and that some of them will have limited scope
206 (regional, facility wide, etc).

207 • *Subscriber Authentication (Core Service-TBD) not yet defined by EPCglobal Architecture*
208 *Framework.*

209 It is important to remember that this is only one view of the EPC Network architecture. Many
210 of the components can be folded into one single piece of software. In certain applications only
211 a few of the roles are actually needed. Even other applications may need components that are
212 either not represented for simplicity (i.e. security) or are not sufficiently developed by the
213 EPCglobal community (i.e. peer to peer based pub/sub event notification).

214 5 ONS Introduction

215 This rest of this document is concerned with specifying the Object Naming Service component
216 discussed above. In keeping with the assumption that the EPCglobal Network architecture
217 should leverage existing Internet standards and infrastructure, ONS uses the Internet’s existing
218 Domain Name System [DNS] for looking up (resolving) information about an EPC. This means
219 that the query and response formats must adhere to the DNS standards, meaning that the EPC
220 will be converted to a domain-name and the results must be a valid DNS Resource Record.

221 **Important terminology note:** the usage of the terms “ONS” and “DNS” may seem arbitrary
222 but they are not. The term DNS is used when the discussion is generally applicable to the DNS
223 system. ONS is used when the discussion is specifically about querying the DNS for an EPC.
224 For example, a query for an MX record is a DNS query. A query to locate an EPC-IS server for
225 an EPC would be called an ONS query, even though the query is carried out using DNS.

226 Additionally, it is important to outline the difference between a “service” and a “server”. A
227 service is a set of functions that accomplish some task. That set of functions may actually be
228 implemented using one or more networked computer systems acting in concert. A good
229 example is the concept of a “Local ONS”, or what is commonly referred to as a local caching
230 name service. In most situations this service is provided by two physically separate servers that
231 act as backups for one another. For information on how to build highly reliable DNS services
232 the reader is directly at numerous industry texts on robust network design.

233 5.1 The Domain Name System (DNS) (non-normative)

234 While not normative, this section is for readers who may be more familiar with systems such as
235 EDI or UCCnet than basic Internet systems that are normally ‘invisible’ to even the most
236 Internet savvy user.

237 In order to correctly understand DNS and how it is used it is important to understand the system
238 from two important viewpoints. The first is from the viewpoint of the client application that is

239 querying DNS. The second is the viewpoint of an entity publishing data into DNS to be used by
240 a client.

241 **5.1.1 Client's View**

242 From a client's standpoint the DNS is a black box. Questions go in and answers come out.
243 Where the answer came from, how long it can be cached, and what sequence of delegations it
244 took to find the answer are all hidden from the client application that issued the query. It is
245 considered an extremely dangerous and potentially disastrous thing to attempt to be smarter
246 than the existing DNS implementations.

247 The practical consequence of this is that the DNS API that client applications will use to
248 perform an ONS query is actually very simple. There are only three pieces of information
249 needed: the nameserver to ask, the domain-name in question, and the type of record that is
250 being requested. The first item, the nameserver to ask, is configured as part of the computer's
251 basic network configuration and usually is not something the client developer should concern
252 themselves with. For example, on a computer that has a properly configured network, the
253 following snippet of Java code is all that is required to issue a DNS query and retrieve the
254 results:

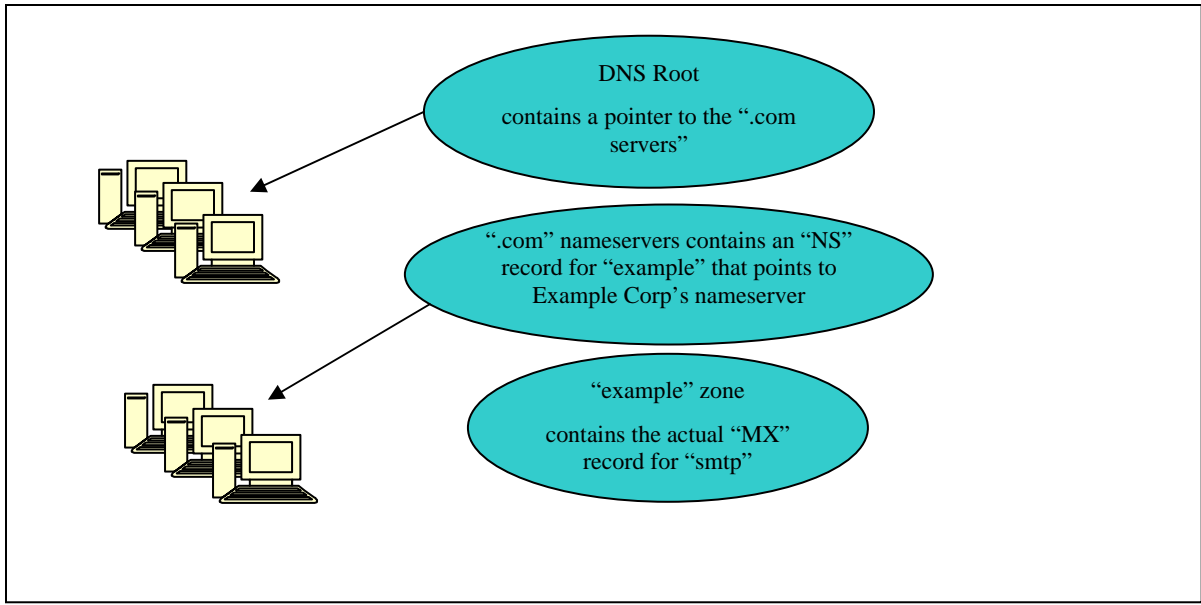
```
DirContext ictx = new InitialDirContext();  
Attributes attrs = ictx.getAttributes("dns://smtp.example.com", "MX");
```

255

256 The “smtp.example.com” is the domain-name and the “MX” is the record type that is being
257 requested. When done the “attrs” variable will contain the hostname for the MX server for the
258 “smtp.example.com” domain. All of the various DNS issues of root servers, caches, secondary
259 servers, etc., are hidden from the client application.

260 **5.1.2 Publisher's View**

261 While client applications are happily able to ignore how the DNS infrastructure actually works,
262 the data the client is consuming must be provisioned by someone in a way that is scalable,
263 secure and accurate. DNS is a system of hierarchically organized servers that roughly follow
264 the hierarchy found in the domain-name. The top of this hierarchy is the DNS Root, often
265 referred to as simply “.” or “dot”. Entries in the root are called “top level domains” or TLDs
266 such as “com”, “net”, “kr”, “us”, etc. For each delegation, or point at which there is a “dot” in a
267 domain-name, there is a delegation to domains lower in the hierarchy. Generally speaking, for
268 each delegation there is a corresponding network server that contains data for that subsection of
269 the hierarchy. This is why DNS is called a “distributed network database”. In the previous
270 example of “smtp.example.com” the delegation of servers would look like this:



271

272 "zone" is considered to be the data that a nameserver publishes. In many situations a single
 273 nameserver can contain multiple zones but it can still be thought of as just a physical collection
 274 of logical 'nameservers'. If the example were for "smtp.department.region.example.com" then
 275 the hierarchy of nameservers would naturally be extended to show these further levels of the
 276 hierarchy.

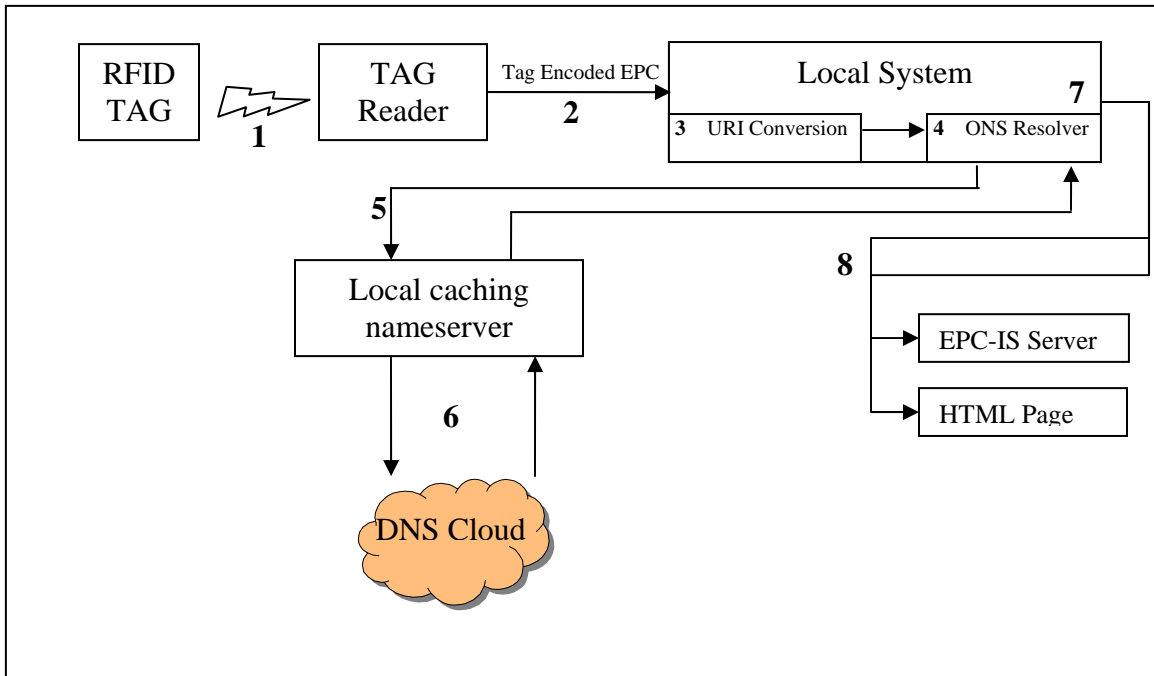
277 This hierarchy of nameservers is used to make the data available. In addition to these servers
 278 there are what are called "caching nameservers". In many cases the caching nameserver and an
 279 enterprises public nameserver are the same thing but logically they're separate functions. A
 280 caching nameserver creates network efficiencies by keeping commonly retrieved records as
 281 close to the querying client as possible. Some operating systems (generally Unix variants) have
 282 caching nameservers built in. But most desktop operating systems offload this responsibility
 283 onto a departmental or enterprise wide caching nameservers. Generally speaking, most
 284 operating systems discover this network configuration information via DHCP.

285 5.2 ONS's Usage of DNS

286 In order to use DNS to find information about an item, the item's EPC must be converted into a
 287 format that DNS can understand, which is the typical, "dot" delimited, left to right form of all
 288 domain-names. The ONS resolution process requires that the EPC being asked about is in its
 289 pure identity URI form as defined by the EPCglobal Tag Data Standard [EPC] (e.g.,
 290 urn:epc:id:sgtin:0614141.100734.1245).

291 Since ONS contains pointers to services, a simple A record (or IP address) is insufficient for
 292 today's more advanced web services based systems. Therefore ONS uses the Naming Authority
 293 PoinTeR (or NAPTR) DNS record type. This record type contains several fields for denoting
 294 the protocol, services and features that a given service endpoint exposes. It also allows the
 295 service end point to be expressed as a URI, thus allowing complex services to be encoded in a
 296 standard way.

297 Figure 3 describes a typical ONS query from start to finish from the viewpoint of a client
 298 application:



299

300

301

Figure 3: A typical ONS query.

302

1. A sequence of bits denoting an EPC is read from a 64-bit RFID tag. Example:

303

(10 000 00000000000000 00000000000000011000 000000000000000110010000)

304

2. The tag Reader sends that sequence of bits to a local server. Example:

305

(10 000 00000000000000 00000000000000011000 000000000000000110010000)

306

3. The local server converts the bit sequence into the pure identity URI Form as defined in Section 4.3.3 of the EPCglobal Tag Data Standards [EPC]. Example:

307

`urn:epc:id:sgtin:0614141.000024.400`

308

309

4. The local server presents the URI to the local ONS Resolver. Example:

310

`urn:epc:id:sgtin:0614141.000024.400`

311

5. The resolver converts the URI form into a domain-name and issues a DNS query for NAPTR records for that domain. Example:

312

`000024.0614141.sgtin.id.onsep.com`

313

314

6. The DNS infrastructure returns a series of answers that contain URLs that point to one or more services (for example, an EPCIS Server). (See Section 10 for examples.)

315

316

7. The local resolver extracts the URL from the DNS record and presents it back to the local server. Example:

317

`http://epc-is.example.com/epc-wsdl.xml`

318

319

8. The local server contacts the correct EPC-IS server found in the URL for the EPC in question

320

321 *Future Note (non-normative): It is expected that the work of the EPCglobal Tag Data*
322 *Translation Working Group, when complete, will provide both a formal representation of the*
323 *transformation procedure above, as well as automated software procedures to carry it out.*

324 **5.2.1 Serial Number Level Queries to the ONS**

325 It is important to note that this version of ONS does not specify queries for fully serialized
326 SGTIN. **It specifically stops at the “Object Class” level.** Subsequent queries for information
327 about a given serial number must be resolved by querying the application layer server
328 designated by the ONS results. The same is true for other forms of EPC including SSCC,
329 GRAI, GIAI, etc. The ability to specify an ONS query at the serial number level of granularity
330 as well as the architectural and economic impacts of that capability is an open issue that will be
331 addressed in subsequent versions of this document. Its lack of mention here should not be
332 construed as making that behavior legal or illegal.

333 **6 ONS Nameserver Infrastructure Organization (non-** 334 **normative)**

335 The previous section covered DNS and how ONS uses DNS from a client point of view. What
336 it did not cover was the processes and procedures for making ONS data available in the proper
337 form for the client. The main issues are the delegation hierarchy and zone maintenance.

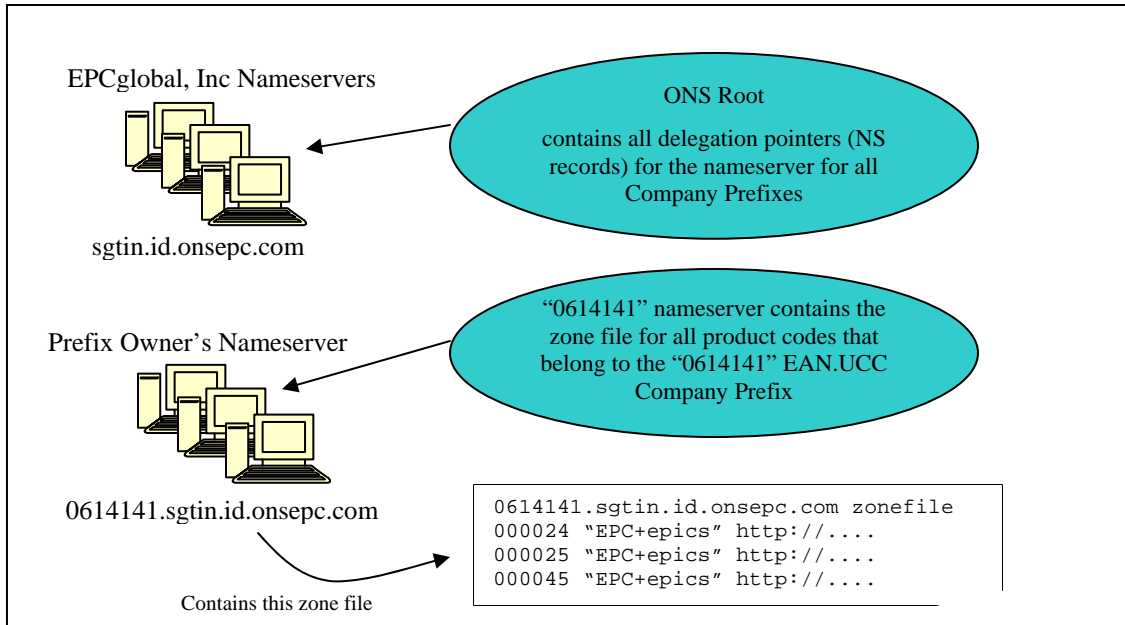
338 **6.1 ONS Delegation Rules**

339 Since ONS is specifically for looking up EPC related services it must follow the delegation
340 rules for the various namespaces that may be part of an EPC. For the sake of discussion below,
341 the SGTIN form of EPC is used as an example. Similar principles apply to the other forms of
342 EPC.

343 An SGTIN is a serialized version of a GTIN. GTINs are part of the EAN.UCC System of
344 product codes. In most of the namespaces that are part of this System there is the concept of an
345 EAN.UCC Company Prefix. Company Prefixes are generally assigned by either EAN or UCC
346 depending on the country the registrant in question belongs to. As part of the deployment of the
347 EPCglobal Network, EAN and UCC have formed a joint venture to handle the various
348 management duties of the network, one of which is coordinating the assignment of Company
349 Prefixes and the provisioning of those prefixes in ONS.

350 Remembering from above that where there is a “dot” there is a delegation step, the “dot” that
351 exists between the product code, the company prefix, and “sgtin.id.onsepc.com” means that
352 there is a delegation step, and thus a pointer to a subsequent zone (and possibly a separate
353 nameserver).

354



355

356 Thus, in order for a company to provision their GTINs with ONS, they must request that this be
 357 done through either UCC or the appropriate EAN affiliate. The request is sent to EPCglobal,
 358 Inc which follows its procedures for determining the contents of the NS record that will appear
 359 in the “sgtin.id.onsepc.com” zone. That NS record will contain the host names of the
 360 nameservers that the owner of that Company Prefix has specified. Once these host names are
 361 available in the “sgtin.id.onsepc.com” zone queries will be routed to the Company Prefix
 362 owner’s nameservers and at that point they can determine which Item Reference codes to
 363 publish information about.

364 6.2 Zone Maintenance Guidelines

365 There are numerous operational guidelines for maintaining DNS nameservers that are either
 366 freely available or come with commercial DNS software. Many of these are specific to the
 367 BIND derived lineage of nameservers but the guidelines are generally applicable to any
 368 nameserver. DNS server software breaks down to two general categories: servers that maintain
 369 their zone data as text based configuration files and those that use sophisticated backend
 370 databases.

371 Many nameservers can synthesize records based on business rules rather than configuration
 372 files. Some companies may have complicated structure within their product codes in order to
 373 segregate by division or region. In such cases nameserver administrators may utilize
 374 “synthesized records” to determine the actual values returned. Whether or not a nameserver
 375 synthesizes its answers from business rules or reads them directly from a text file is irrelevant
 376 to how ONS works and is a hidden, server-side optimization.

377 7 ONS Formal Specification

378 The formal specification of ONS is a set of procedures and rules to be followed by ONS Clients
 379 and ONS Publishers. An ONS Client is an application that wishes to use ONS to identify a
 380 service that may provide information about a specific EPC. An ONS Publisher is an entity
 381 responsible for making services available to ONS Clients by creating service pointer entries in

382 an ONS Server. An ONS Server is an implementation of a DNS Server; because ONS differs
383 from DNS only in what data is provided by the server, not in the operation of the server itself,
384 there is no separate specification for an ONS Server. Any DNS server compliant with [DNS]
385 and [RFC 3403] may be used as an ONS Server.

386 The ONS specification consists of three ingredients:

- 387 • A procedure (Section 8) that an ONS Client SHALL follow in order to present a query to
388 ONS. This procedure specifies how an EPC is converted to a DNS NAPTR query.
- 389 • A set of rules (Section 9) that ONS Publishers SHALL follow to represent ONS information
390 (namely, pointers to services for EPCs) as DNS NAPTR records within an ONS Server.
- 391 • A procedure (Section 10) that an ONS Client SHALL follow in order to interpret the
392 results of an ONS query. This procedure specifies how an ONS Client can locate a service
393 using the information provided by the ONS Server.

394 **8 DNS Query Format**

395 The following specifies the procedure an ONS client SHALL follow to present a query to ONS.

- 396 1. Begin with an EPC represented in the pure identity URI form as defined in Section 4.1 of
397 the EPCglobal Tag Data Standards [EPC]. URIs in this form are ASCII strings beginning
398 with `urn:epc:id:`, for example, `urn:epc:id:sgtin:0614141.000024.400`.
- 399 2. Follow the procedure below to convert the URI into a domain name ending with
400 `.onsepc.com`. For example, `000024.0614141.sgtin.id.onsepc.com`.
- 401 3. Use a DNS resolver to query for DNS Type Code 35 (NAPTR) records for the domain
402 name from Step 2. The method for obtaining and using the DNS resolver is outside the
403 scope of this specification. It is anticipated that this will be addressed through a companion
404 specification that specifies a standard ONS Application Programming Interface (API).
405 Even when a standard ONS API exists, however, an ONS Client MAY use any DNS
406 resolver conforming to [DNS], using whatever API is available.
- 407 4. Follow the procedure in Section 10 to interpret the results from Step 3.

408 In order to query the DNS for the EPC, the URI form specified above must be converted to
409 domain name form in Step 2. The procedure for this conversion is as follows:

- 410 1. Begin with an EPC represented in the pure identity URI form as defined in Section 4.3.3 of
411 the EPCglobal Tag Data Standards [EPC]. For example,
412 `urn:epc:id:sgtin:0614141.000024.400`.
- 413 2. Remove the `urn:epc:` prefix (in the example, leaving
414 `id:sgtin:0614141.000024.400`).
- 415 3. Remove the serial number field. In all tag formats currently defined in [EPC] (SGTIN,
416 SSCC, SGLN, GRAI, GIAI, and GID), the serial number field is the rightmost period (.)
417 character and all characters to the right of it. (In the example, this leaves
418 `id:sgtin:0614141.000024`)
- 419 4. Replace each colon (:) character with a period (.) character (in the example, leaving
420 `id.sgtin.0614141.000024`)

- 421 5. Invert the order of the remaining period-delimited fields (in the example, leaving
 422 000024.0614141.sgtin.id)
- 423 6. Append .onsepc.com. In the example, the result is
 424 000024.0614141.sgtin.id.onsepc.com.

425 9 DNS Records for ONS

426 ONS contains pointers to authoritative information for EPCs. Each such pointer takes the form
 427 of a DNS Type Code 35 (NAPTR) record. This section specifies how ONS Publishers must
 428 encode information into NAPTR records.

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

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

430

431 ONS Publishers SHALL obey the following rules:

- 432 • The **Order** field SHALL be zero.

433 *Explanation (non-normative): The Order field is used in DNS applications where a series of*
 434 *regular expressions from distinct NAPTR records are applied consecutively to an input. ONS*
 435 *does not **currently** make use of this feature but future requirements may force a re-evaluation of*
 436 *always specifying this number as zero. Implementers are strongly encouraged to follow the*
 437 *DDDS algorithm as specified in [RFC 3401].*

- 438 • The **Pref** field SHALL be a non-negative integer. The value of the Pref field is an ordinal
 439 that specifies that the service in one record is preferred to the service in another record
 440 having the same Service field. An ONS Client SHOULD use attempt to use a service
 441 having a lower Pref number before using an equivalent service having a higher Pref
 442 number.
- 443 • The **Flags** field SHALL be set to 'u', indicating that the **Regexp** field contains a URI.
- 444 • The **Service** field contains an indicator of the type of service that can be found at the URI in
 445 question. This feature allows for the ONS service to indicate different service end points for
 446 different classes of service. The value of the Service field SHALL consist of the string
 447 EPC+ followed by the name of a service registered with the EPC Network Protocol
 448 Parameter Registry. The Service field identifies what type of service is accessible through
 449 the URL provided by the Regexp field of this record. See Section 10 for examples of
 450 service types, and Section 15 for a list of services that make up an initial set of registered
 451 service classes.
- 452 • The **Replacement** field is not used by the EPC Network but since it is a special DNS field
 453 its value is set to a single period ('.') instead of simply a blank.
- 454 • The **Regexp** field specifies a URL for the service being described. The value of this field
 455 SHALL be the string !^.*\$! (the six character sequence consisting of an exclamation
 456 point, a caret, a period, an asterisk, a dollar sign, and another exclamation point), followed
 457 by a URL, followed by an exclamation point (!) character.

458 *Explanation (non-normative): In previous versions of ONS the result was simply an IP*
459 *address. This proved insufficient due to the needs of protocols such as SOAP that are layered*
460 *over HTTP. In nearly all modern protocols there is a need for a hostname and additional 'path'*
461 *information. The reason the field is in the form of a regular expression is that the NAPTR*
462 *record is used by other applications that have the need to conditionally rewrite the URI to*
463 *include other information. While none of the examples here make use of this feature, it has not*
464 *been determined if this will always be the case. In the future it may become necessary to allow*
465 *full regular expression and replacement functions within the regexp field. Therefore,*
466 *implementers would be wise to not assume that the URI can simply be extracted without any*
467 *regular expression processing.*

468 *The general form of the Regexp field is as a Posix Extended Regular Expression. This form*
469 *states that the first character encountered is the field delimiter between the regular expression*
470 *and the replacement portion of the entire rewrite expression. In the above example the delimiter*
471 *is the exclamation point (!) character. The regular expression portion is `^.*$` which equates*
472 *to 'match anything'. The replacement portion is `http://example.com/cgi-`*
473 *bin/epcis. The choice of '!' as the delimiter instead of a more traditional '/' makes the*
474 *entire line much easier to read and less error prone.*

475 *In a future version of this specification, if regular expression processing becomes necessary,*
476 *the input to the regular expression processor would be the original canonical EPC URI, before*
477 *transforming to an `onsepc.com` domain name.*

478 **10 Processing ONS Query Responses**

479 ONS Clients SHALL use the following procedure to interpret the results returned by an ONS
480 query as formulated in Section 8.

- 481 1. The result from the ONS query is a set of NAPTR records as described in Section 9.
- 482 2. From among the results from Step 1, select those records whose Service field names the
483 desired service. If there is no such record, stop: a pointer to the desired service is not
484 available from ONS for the specified EPC.
- 485 3. From among the results from Step 2, select those records having the lowest value in the Pref
486 field.
- 487 4. From among the results from Step 3, select a record at random.
- 488 5. Extract the service URL from the record from Step 4, by extracting the substring between
489 the initial `!^.*$!` and the final `!` character.
- 490 6. Attempt to use the service URL from Step 5.
- 491 7. If Step 6 is not successful, go back to Step 4, using a different record from among the
492 records from Step 3. If all records from Step 3 have been tried, go back to Step 3 using
493 records from Step 2 having the next lowest value in the Pref field. If all records from Step 2
494 have been tried, stop: no service is available.

495 **11 Examples (non-normative)**

496 In the following examples the EPC in question is `urn:epc:id:sgtin:0614141.011015.583865`
497 which represents an Example Corporation Model 100 Widget.

498 The ONS Client application attempts to learn about this product by first following the
 499 procedure in Section 8, which converts the EPC into a domain-name:
 500 011015.0614141.sgtin.id.onsepc.com

501 The application then queries the DNS for NAPTR records for that domain name and receives
 502 the following records:

Order	Pref	Flags	Service	Regexp	Replacement
0	0	u	EPC+ws	!^.*\$!http://example.com/autoid/widget100.wsdl!	.
0	0	u	EPC+epcis	!^.*\$!http://example.com/autoid/cgi-bin/epcis.php!	.
0	0	u	EPC+html	!^.*\$!http://www.example.com/products/thingies.asp!	.
0	0	u	EPC+xmlrpc	!^.*\$!http://gateway1.xmlrpc.com/servlet/example.com!	.
0	1	u	EPC+xmlrpc	!^.*\$!http://gateway2.xmlrpc.com/servlet/example.com!	.

503

504 Each of these records conforms to the rules specified in Section 9.

505 Finally, depending on the service that the ONS Client desires, it uses one or more of the records
 506 returned to locate an appropriate service. The following sections describe specific examples of
 507 services an ONS Client might locate. In each case, the ONS Client uses the procedure specified
 508 in Section 10 to locate a service, using the records returned above.

509 **11.1 Finding a WSDL file for a product**

510 One of the simplest but most powerful examples is where an ERP application is Web Services
 511 [Web Services] enabled. This particular application is capable of learning about new products
 512 by making various application specific web services calls to public interfaces made available by
 513 the manufacturer. In this case the application can simply use the ONS to look to see if a WSDL
 514 file exists that describes the Web Services it requires.

515 The application issues the query and receives the NAPTR records above. It iterates through the
 516 list looking for the 'ws' service which designates a WSDL file that defines the available web
 517 services. It locates that service in the first record and returns the URI found in the Regexp field.
 518 The application then hands that URI to its Web Services engine to determine if the proper end
 519 points exist. If they do then the application requests the metadata it needs and proceeds with its
 520 processing.

521 **11.2 Finding an authoritative EPCIS server for a product**

522 This example shows how an EPC can be used to retrieve a pointer to an EPCIS server that
 523 contains authoritative metadata about a product. Again, using the same results from above, the
 524 client uses the second record and extracts the URI from the Regexp. It then uses that URI as the
 525 end point to send the EPCIS query to.

526 **11.3 Finding an HTML formatted web page description of a product**

527 This example shows how a very lightweight service can be deployed almost immediately using
 528 nothing more than an existing externally available corporate web server containing existing
 529 product content. By inserting the third record in the results list above, Example Corp can easily

530 point applications to authoritative product data without any modifications to their systems.
531 Applications that understand this service can be very lightweight, using existing web browsers
532 to display the content.

533 **11.4 Finding an XML-RPC gateway to the Web Service interfaces**

534 In some cases a service may be outsourced. In this example Example Corp has decided not to
535 expose an XML-RPC [XML-RPC] service of its own. Instead there is an XML-RPC to SOAP
536 gateway run by a third party. In the interest of interoperability Example Corp simply adds an
537 ONS entry that points to this gateway, thus enabling new applications with little effort on their
538 part.

539 In this case there are two records for this service and both have the same Order value. This
540 means that the Pref field is used to indicate a preference for one over the other (load balancing
541 and fail-over). If for some reason the record with the Pref of '0' fails or is busy, the one with the
542 Pref of '1' can be used.

543 **12 References**

544 **EPCAF**

545 K. R. Traub et al, "EPCglobal Architecture Framework," EPCglobal technical
546 document, July 2005. (See <http://www.epcglobalinc.org/standards/architecture>)

547 **Willinger and Doyle**

548 Willinger, W. and Doyle, John. *Robustness and the Internet: Design and evolution*,
549 2002. (See http://netlab.caltech.edu/pub/papers/part1_vers4.pdf)

550 **EPC**

551 EPCglobal, "EPCglobal Tag Data Standards Version 1.3.1," EPCglobal Ratified
552 Standard, September 2007. (See <http://www.epcglobalinc.org/standards/tds>)

553 **DNS**

554 (Internet Engineering Task Force). STD0013, *RFC 1034, RFC 1035*, ed Mockapetris, P.
555 2000. (See <http://www.ietf.org/rfc/std/std13.txt>)

556 **Web Services**

557 (WorldWideWeb Consortium). *Web Services Activity*, 2000. (See
558 <http://www.w3.org/2002/ws/>)

559 **XML-RPC**

560 Winer, Dave. *XML-RPC Specification*, 1999. (See <http://www.xml-rpc.com/spec>)

561 **RFC 2396**

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

564 **RFC 3401**

565 Mealling, Michael. *Dynamic Delegation Discovery System (DDDS) Part One: The*
566 *Comprehensive DDDS*, 2002. (See <http://www.ietf.org/rfc/rfc3401.txt>)

567 **RFC 3403**

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

570 **13 Appendix A – Glossary (non-normative)**

571 **Auto-ID**

572 "Automatic Identification" -- An open, global network that can identify anything,
573 anywhere, automatically.

574 **Domain-name**

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

576 **DNS**

577 See Domain Name Service

578 **Domain Name Service**

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

583 **EPC**

584 See Electronic Product Code

585 **EPCIS**

586 EPC Information Service – A series of EPC Network specific standards defining various
587 methods of data exchange and metadata lookup.

588 **Electronic Product Code**

589 An abstract namespace made up of an EPC Manager Number, an Object Class code, and
590 a Serial Number, or a subset thereof.

591 **EPC Manager Number**

592 A code that identifies a manufacturer of objects

593 **ONS**

594 See Object Name Service

595 **ONS Root**

596 The domain-name that is appended to the manager id and which acts as the 'top' of the
597 DNS tree that contains all of the EPC entries. This root domain is "onsepc.com"

598 **Object Name Service**

599 A resolution system, based on DNS, for discovering authoritative information about an
600 EPC

601

602 **Object Class code**

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

605 **NAPTR**

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

609 **Reader**

610 A radio enabled device that communicates with a tag

611 **RFID**

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

616 **Regular Expression**

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

620 **Serial Number**

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

622 **tag**

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

625 **URI**

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

629 **URL**

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

633 **14 Appendix B -- DDDS Application Specification (non-**
634 **normative)**

635 The use of NAPTR records is governed by a series of RFCs that define something called the
636 Dynamic Delegation Discovery Service. RFC 3401 [RFC 3401] is the first in the series and
637 provides an introduction to the series. In order to safely use NAPTR records on the public
638 network a specification must exist that describes the values of the various fields. This appendix
639 contains that specification which, when approved by the SAG process, will be extracted and
640 published as an RFC itself.

641 **14.1 Application Unique String**

642 The Application Unique String is the EPC in URI form.

643 **14.2 First Well Known Rule**

644 The First Well Known Rule is the identity function. The output of this rule is the same as the
645 input. This is because the EPC namespace and this Applications databases are organized in such
646 a way that it is possible to go directly from the name to the smallest granularity of the
647 namespace directly from the name itself.

648 **14.3 Expected Output**

649 The output of the last Rewrite Rule is a URI and a Service designator that, together, designate
650 an application context (server and application) that will expose some metadata or services about
651 the EPC.

652 **14.4 Valid Databases**

653 At present only one DDDS Database is specified for this Application. RFC 3403 [RFC 3403]
654 specifies a DDDS Database that uses the NAPTR DNS resource record to contain the rewrite
655 rules. The Keys for this database are encoded as domain-names. The conversion method for this
656 database is as follows:

- 657 1. Remove the 'urn:epc:' header
658 2. Remove the serial number field
659 3. Invert the order of the remaining fields
660 4. Convert all ':' characters to '.'
661 5. Append ".onsepc.com"

662 **14.5 Valid Flags**

663 The 'u' flag which denotes that the current Rule is terminal and that the output of the Regexp is
664 a URI.

665 **14.6 Service Parameters**

666 The Service parameters for this Application take the form of a string of characters with the
667 following ABNF:

668 service_field = "EPC+" service_name
669 service_name = ALPHA *31ALPHANUM

670 The valid values for 'service_name' and the process for registering new services are found in the
671 ONS Service Registry discussed in Appendix C. It is important to note that the "+" character is
672 not allowed in a service_name. This allows for future expansion of the service field if it
673 becomes apparently that service_names may need parameters.

674 **15 Appendix C -- Service Field Registrations (non-** 675 **normative)**

676 The 'service_name' portion of the Service field is a managed space where the values that are
677 available for use are found in the EPC Network Protocol Parameters Registry. The goal is to
678 balance the ability to innovate with new services with the desire for interoperability between
679 services. If the service_name simply ends up denoting non-interoperable, proprietary services
680 then the total value of the system is significantly reduced. Conversely, limiting all services to
681 simply those that are standardized by the EPCglobal SAG standards process limits the ability of
682 the system to innovate.

683 To balance these requirements a special sub-class of services is created which start with an 'x-'.
684 Services of this type do not need to be registered but merely act as a designation of
685 "experimental status". Implementers should not create services in this class and then never
686 register them. This method has been utilized within the MIME Content-Type registry for years
687 and while there are instances of "x-" entries becoming common, this is becoming less common
688 as the process is streamlined.

689 The Registry entry for ONS Service types will be created with the /ons/service_name/
690 pathname. Files in that directory will be named after Service Name field in the template
691 followed by the '.txt' file extension. The contents of the file will be the template supplied for
692 the registration. For example, the 'epcis' template below would be found at
693 http://onsepc.com/ons/service_name/epcis.txt and would contain the just
694 the fields mentioned below. Registrations are First Come First Served. Registration requires a
695 published Recommendation from EPCglobal.

696 **15.1 Registration Template**

697 A document specifying a service_name must contain the following template. This template is
698 then entered into the registry as soon as the document is published.

699 **Service Name:**

700 The exact sequence of characters that will appear in the Service field

701 **Functional Specification:**

702 A pointer to publicly available documentation for the service.

703 **Valid URI schemes:**

704 A list of registered URI schemes that are valid for this service (e.g. web services can be
705 specified using either the 'http:' or 'https:' scheme)

706 **Security Considerations:**

707 Any security issues associated with use of this service

708 **15.2 Service Registrations**

709 The following are initial services that will be registered as soon as this document is published
710 as a specification:

- 711 • **Service Name: epcis**

712 **Functional Specification:** The EPC Information Service Specification
713 **Valid URI schemes:** http, https
714 **Security Considerations:** See the Security Considerations section of the EPC Information
715 Service Specification

716 • **Service Name: ws**

717 **Functional Specification:** A generic Web Services [Web Services] based service where the
718 application must negotiate what services are available by investigating the WSDL file found
719 at the URI in the Regexp
720 **Valid URI schemes:** http, https
721 **Security Considerations:** Web Services utilize a great deal of existing Internet
722 infrastructure and protocols. It is very easy to use some of them in insecure ways. Any
723 usage of Web Services should be done in the context of a thorough understanding of the
724 dependencies, especially as it relates to the DNS and HTTP.

725 • **Service Name: html**

726 **Functional Specification:** Simply returns a URI that will resolve to an HTML page on
727 some server. The assumption is that this page contains information about the product in
728 question.
729 **Valid URI schemes:** http, https, ftp
730 **Security Considerations:** None not already inherent in the use of the WorldWideWeb.

731 • **Service Name: xmlrpc**

732 **Functional Specification:** A URI that denotes an HTTP POST capable service on some
733 server that is expecting XML-RPC [XML-RPC] compliant connection.
734 **Valid URI schemes:** http, https
735 **Security Considerations:** None not already inherent in the use of the WorldWideWeb.
736