

1  
2

3



4

Object Naming Service (ONS)  
Version 1.0

5

6

7

8

9

10

EPCglobal Ratified Specification  
Version of October 4, 2005

11

12

13 **Abstract**

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

18 **Status of this document**

19 This section describes the status of this document at the time of its publication. Other  
20 documents may supersede this document. The latest status of this document series is  
21 maintained at EPCglobal. This document was ratified by the EPCglobal Board of  
22 Governors on October 4, 2005.

23 Comments on this document should be sent to the EPCglobal Software Action Group  
24 (SAG) mailing list ([sag\\_ons@epclinklist.epcglobalinc.org](mailto:sag_ons@epclinklist.epcglobalinc.org)).

25

26

27

28

29

30

## 31 **Table of contents**

32	1	Introduction.....	5
33	2	Terminology and Typographical Conventions .....	5
34	3	Background Information (non-normative) .....	5
35	4	EPCglobal Network Architecture (non-normative).....	5
36	4.1	Electronic Product Code (EPC) .....	6
37	4.2	EPCglobal Network Software Architecture Components .....	6
38	5	Object Naming Service (ONS) Introduction.....	9
39	5.1	The Domain Name System (DNS) (non-normative).....	10
40	5.1.1	Client's View .....	10
41	5.1.2	Publisher's View .....	10
42	5.2	ONS's Usage of DNS.....	11
43	5.2.1	Serial Number Level Queries to the ONS.....	13
44	6	ONS Nameserver Infrastructure Organization (non-normative) .....	13
45	6.1	ONS Delegation Rules .....	13
46	6.2	Zone Maintenance Guidelines .....	14
47	7	ONS Formal Specification .....	14
48	8	DNS Query Format.....	15
49	9	DNS Records for ONS.....	16
50	10	Processing ONS Query Responses.....	17
51	11	Examples (non-normative) .....	18
52	11.1	Finding a WSDL file for a product.....	18
53	11.2	Finding an authoritative EPCIS server for a product .....	19
54	11.3	Finding an HTML formatted web page description of a product .....	19
55	11.4	Finding an XML-RPC gateway to the Web Service interfaces .....	19
56	12	References.....	19
57	13	Appendix A – Glossary (non-normative) .....	20
58	14	Appendix B -- DDDS Application Specification (non-normative) .....	22
59	14.1	Application Unique String .....	22
60	14.2	First Well Known Rule .....	22
61	14.3	Expected Output.....	22
62	14.4	Valid Databases.....	22
63	14.5	Valid Flags .....	23
64	14.6	Service Parameters .....	23

65 15 Appendix C -- Service Field Registrations (non-normative) ..... 23  
66 15.1 Registration Template..... 23  
67 15.2 Service Registrations..... 24  
68  
69

## 70 **1 Introduction**

71 This document specifies how the Domain Name System is used to locate authoritative  
72 metadata and services associated with the SGTIN portion of a given Electronic Product  
73 Code™ (EPC). Its target audience is developers that will be implementing Object  
74 Naming Service (ONS) resolution systems for applications. Future work by the ONS  
75 Working Group will address how ONS is used for the other namespaces that make up  
76 the EPC and that are outlined in the EPCglobal Tag Data Standard  
77 [\[Tag Data Standards\]](#).

## 78 **2 Terminology and Typographical Conventions**

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

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

86 The following typographical conventions are used throughout the document:

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

## 93 **3 Background Information (non-normative)**

94 This document draws from the previous work at the Auto-ID Center, and we would like to  
95 recognize the contributions of the following individuals: Joe Foley (MIT), Erik Nygren  
96 (MIT), Sanjay Sarma (MIT), David Brock (MIT), Sunny Siu (MIT), Laxmiprasad Putta  
97 (OATSystems), Sridhar Ramachandran (OATSystems). The following papers capture  
98 the contributions of these individuals:

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

## 104 **4 EPCglobal Network Architecture (non-normative)**

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

## 111 **4.1 Electronic Product Code (EPC)**

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

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

127 Each of the sub-namespaces that are defined by the Tag Data Standard have a slightly  
128 different structure depending on what they identify, how they are used, and how they are  
129 assigned. The SGTIN is used to identify an individual product that is assigned by the  
130 company that creates that product. Thus the SGTIN contains an EPC Manager Number,  
131 an Object Class, and a Serial Number. Other sub-namespaces such as the SSCC go  
132 directly from the EPC Manager Number to the Serial Number and have no concept of an  
133 "Object Class". This document only specifies the ONS lookup mechanism for the SGTIN  
134 sub-namespace. As such, in many cases its statements about concepts such as "Object  
135 Class" or "Serial Number" are specific to the SGTIN namespace and should not be  
136 construed as applying to all EPC namespaces. Specifications for those other  
137 namespaces are the subject of future work within the ONS Working Group.

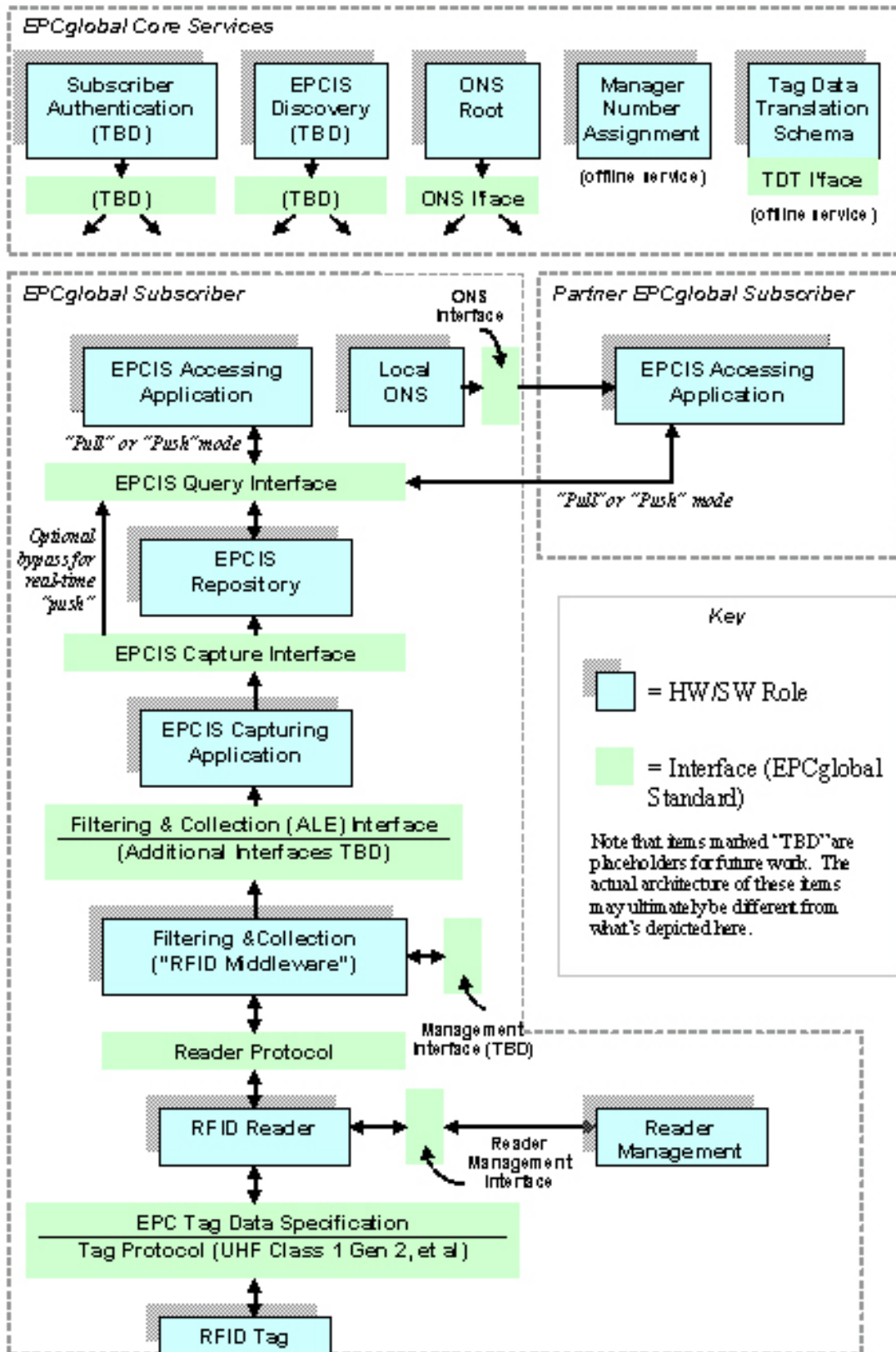
138 In order to further leverage the use of Internet derived technology and systems, the EPC  
139 is encoded as a Uniform Resource Identifier (URI). URIs are the basic addressing  
140 scheme for the entire World Wide Web and ensure that the EPC Network is compatible  
141 with the Internet going forward.

142 While an addressing scheme by itself is useful, it can only be used within a network  
143 when a mechanism is provided to **authoritatively** look up information about that  
144 identifier [\[RFC 2826\]](#). This EPC 'resolution' mechanism is called the Object Naming  
145 Service, or ONS and is what forms the core integrating, or 'truth' verifying, principle of  
146 the EPCglobal Network.

## 147 **4.2 EPCglobal Network Software Architecture Components**

148 The EPCglobal Network Architecture as in Fig. 1 shows the high-level components of  
149 the EPCglobal Network. This section highlights and makes reference to The EPCglobal  
150 Network Architecture Framework Version 1, July 2005. Available on the EPCglobal  
151 website at:

152 [http://www.epcglobalinc.org/standards\\_technology/Final-epcglobal-arch-20050701.pdf](http://www.epcglobalinc.org/standards_technology/Final-epcglobal-arch-20050701.pdf)



155

## Figure 1: EPCglobal Network Architecture: Components and Layers

156

Components in blue are “roles”, not discrete pieces of software. Components in green

157

are interfaces used between two roles. The yellow components labeled “ERP”

158

correspond to locations where one or more ERP systems might provide or interface with

159

a particular role. These components from the figures above are described in the sections

160

below:

161

- *Readers*

162

Make multiple observations of RFID tags while they are in the read zone.

163

- *Reader Protocol Interface*

164

Delivers raw tag reads from Readers to Middleware. Events at this interface say

165

“Reader A saw EPC X at time T.”

166

- *Middleware*

167

Accumulates and filters raw tag reads

168

- *ALE Interface*

169

Delivers consolidated, filtered tag read data from Middleware to Local Application.

170

Events at this interface are “At Location L, between time T1 and T2, the following

171

EPCs were observed” where the list of EPCs has no duplicates and has been filtered

172

by appropriate criteria.

173

- *EPC Capturing Application*

174

Recognizes the occurrence of EPC-related business events, and delivers these as

175

EPCIS data

176

- *EPCIS Capture Interface*

177

Provides a path for communicating EPCIS events generated by EPCIS Capturing

178

Applications to other roles that require them, including EPCIS Repositories, internal

179

EPCIS Accessing Applications, and Partner EPCIS Accessing Applications.

180

- *EPCIS Repository*

181

Records EPCIS-level events generated by one or more EPCIS Capturing

182

Applications, and makes them available for later query by EPCIS Accessing

183

Applications.

184

- *EPCIS Query Interface*

185

Provides means whereby an EPCIS Accessing Application can request EPCIS data

186

from an EPCIS Repository or an EPCIS Capturing Application, and the means by

187

which the result is returned.

188

- *EPCIS-Accessing Application*

189

Software that carries out overall enterprise business processes, such as warehouse

190

management, shipping and receiving, historical throughput analysis, and so forth,

191

aided by EPC-related data.

192

- *Local ONS*

193

Fulfills ONS lookup requests for EPCs within the control of the enterprise that

194

operates the Local ONS; that is, EPCs for which the enterprise is the EPC Manager.

195

- *EPCIS Accessing Application*

196

An EPCIS-enabled Application of a trading partner. Partner Applications may be

197

granted access to a subset of the information that is available within the enterprise.

198

- *Tag Data Translation Schema*

199

Provides a machine-readable file that defines how to translate between EPC

200 encodings defined by the EPC Tag Data Specification. EPCglobal provides this file  
201 for use by End-users, so that components of their infrastructure may automatically  
202 become aware of new EPC formats as they are defined.

203 • *Manager Number Assignment*

204 Ensures global uniqueness of EPCs by maintaining uniqueness of EPC Manager  
205 Numbers assigned to EPCglobal Subscribers

206 • *Object Name Service (ONS) Root*

207 A service that, given an EPC, can return a list of network accessible service  
208 endpoints that pertain to the EPC in question. ONS **does not** contain actual data  
209 about the EPC. It only contains the network address of services that contain the  
210 actual data. ONS is also **authoritative** in that the entity that has change control over  
211 the information about the EPC is the same entity that assigned the EPC to the item  
212 to begin with. For example, in the case of an SGTIN EPC, the entity having control  
213 over the ONS record is the owner of the SGTIN manager number (EAN.UCC  
214 Company Prefix).

215 • *EPC Discovery Service(s)*

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

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

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

## 231 5 ONS Introduction

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

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

245 Additionally, it is important to outline the difference between a “service” and a “server”. A  
246 service is a set of functions that accomplish some task. That set of functions may

247 actually be implemented using one or more networked computer systems acting in  
248 concert. A good example is the concept of a “Local ONS”, or what is commonly referred  
249 to as a local caching nameservice. In most situations this service is provided by two  
250 physically separate servers that act as backups for one another. For information on how  
251 to build highly reliable DNS services the reader is directed to numerous industry texts on  
252 robust network design.

## 253 **5.1 The Domain Name System (DNS) (non-normative)**

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

257 In order to correctly understand DNS and how it is used it is important to understand the  
258 system from two important viewpoints. The first is from the viewpoint of the client  
259 application that is querying DNS. The second is the viewpoint of an entity publishing  
260 data into DNS to be used by a client.

### 261 **5.1.1 Client’s View**

262 From a client’s standpoint the DNS is a black box. Questions go in and answers come  
263 out. Where the answer came from, how long it can be cached, and what sequence of  
264 delegations it took to find the answer are all hidden from the client application that issued  
265 the query. Even the concept of being “hidden” isn’t strong enough to convey the concept  
266 since some applications would be tempted to “peek under the hood”. Since DNS  
267 conveys information such as valid Time To Live and where authoritative answers came  
268 from within the actual wire protocol, an application cannot recover enough information by  
269 peeking under the hood to ensure that it is using the correct responses. Therefore it is  
270 considered an extremely dangerous and potentially disastrous thing to attempt to be  
271 smarter than the existing DNS implementations.

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

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

280

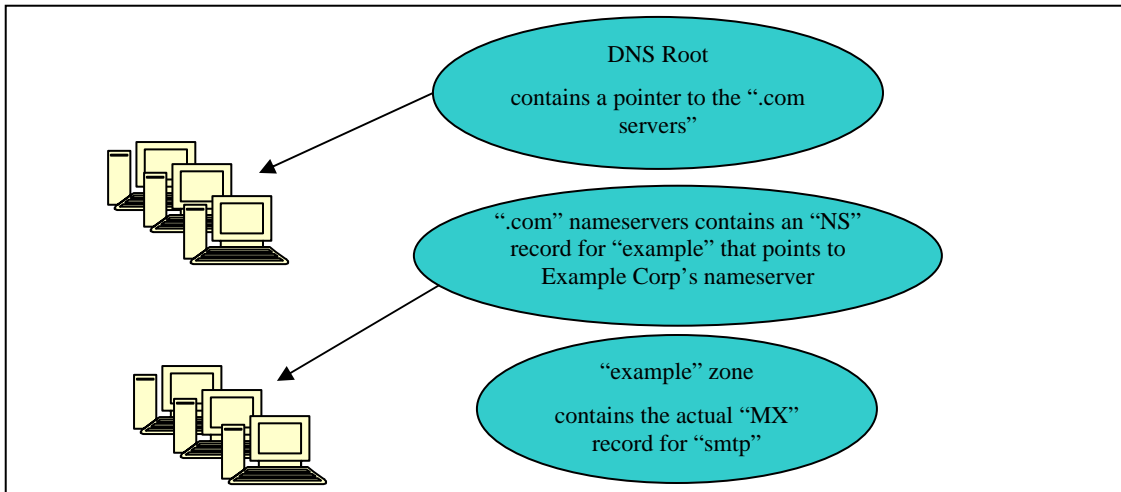
281 The “smtp.example.com” is the domain-name and the “MX” is the record type that is  
282 being requested. When done the “attrs” variable will contain the hostname for the MX  
283 server for the “smtp.example.com” domain. All of the various DNS issues of root servers,  
284 caches, secondaries, etc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

### 285 **5.1.2 Publisher’s View**

286 While client applications are happily able to ignore how the DNS infrastructure actually  
287 works, the data the client is consuming must be provisioned by someone in a way that is

288 scalable, secure and accurate. DNS is a system of hierarchically organized servers that  
289 roughly follow the hierarchy found in the domain-name. The top of this hierarchy is the  
290 DNS Root, often referred to as simply “.” or “dot”. Entries in the root are called “top level  
291 domains” or TLDs such as “com”, “net”, “kr”, “us”, etc. For each delegation, or point at  
292 which there is a “dot” in a domain-name, there is a delegation to domains lower in the  
293 hierarchy. Generally speaking, for each delegation there is a corresponding network  
294 server that contains data for that subsection of the hierarchy. This is why DNS is called a  
295 “distributed network database”. In the previous example of “smtp.example.com” the  
296 delegation of servers would look like this:

297



298

299 A “zone” is considered to be the data that a nameserver publishes. In many situations a  
300 single nameserver can contain multiple zones but it can still be thought of as just a  
301 physical collection of logical ‘nameservers’. If the example were for  
302 “smtp.department.region.example.com” then the hierarchy of nameservers would  
303 naturally be extended to show these further levels of the hierarchy.

304 This hierarchy of nameservers is used to make the data available. In addition to these  
305 servers there are what are called “caching nameservers”. In many cases the caching  
306 nameserver and an enterprises public nameserver are the same thing but logically  
307 they’re separate functions. A caching nameserver creates network efficiencies by  
308 keeping commonly retrieved records as close to the querying client as possible. Some  
309 operating systems (generally Unix variants) have caching nameservers built in. But most  
310 desktop operating systems offload this responsibility onto a departmental or enterprise  
311 wide caching nameservers. Generally speaking, most operating systems discover this  
312 network configuration information via DHCP.

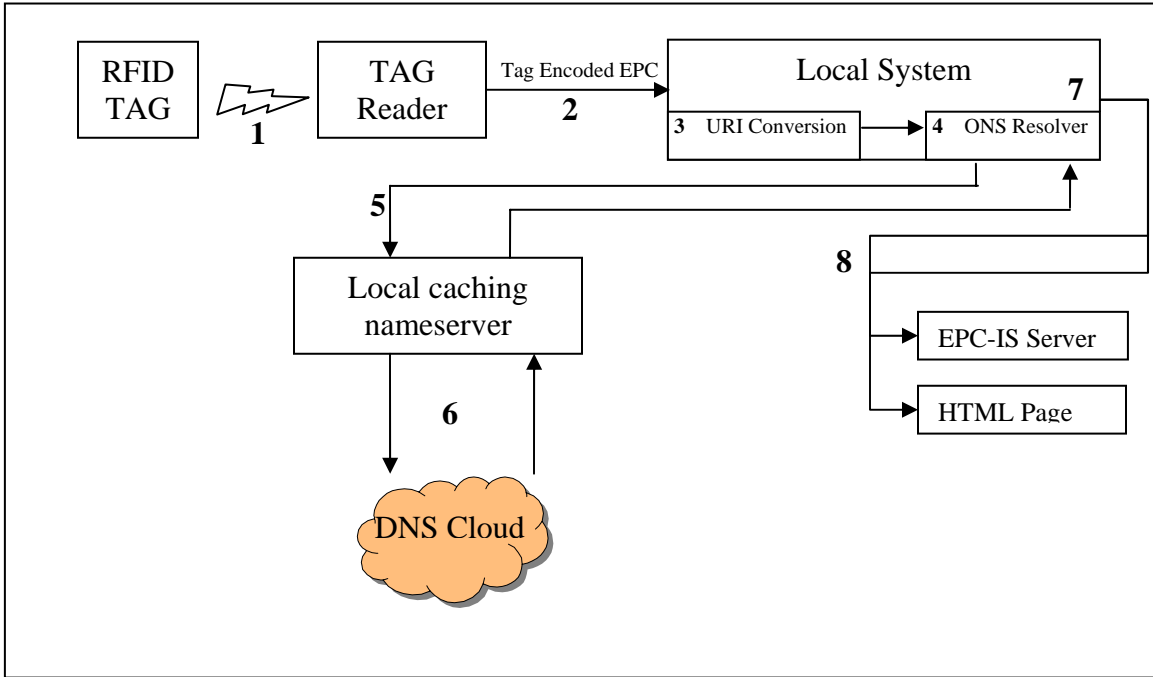
## 313 5.2 ONS’s Usage of DNS

314 In order to use DNS to find information about an item, the item’s EPC must be converted  
315 into a format that DNS can understand, which is the typical, “dot” delimited, left to right  
316 form of all domain-names. The ONS resolution process requires that the EPC being  
317 asked about is in its pure identity URI form as defined by the EPCglobal Tag Data  
318 Standard [\[TAG Data Standard\]](#) (e.g., urn:epc:id:sgtin:0614141.100734.1245).

319 Since ONS contains pointers to services, a simple A record (or IP address) is insufficient  
320 for today’s more advanced web services based systems. Therefore ONS uses the  
321 Naming Authority PoinTeR (or NAPTR) DNS record type. This record type contains

322 several fields for denoting the protocol, services and features that a given service  
 323 endpoint exposes. It also allows the service end point to be expressed as a URI, thus  
 324 allowing complex services to be encoded in a standard way.

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



327  
 328

**Figure 3: A typical ONS query.**

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

Example:

(10 000 00000000000000 00000000000000011000 0000000000000000110010000)

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

(10 000 00000000000000 00000000000000011000 0000000000000000110010000)

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 Standard [TAG Data Standards]. Example:

urn:epc:id:sgtin:0614141.000024.400

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

urn:epc:id:sgtin:0614141.000024.400

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

000024.0614141.sgtin.id.onsepc.com

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.)

344  
 345  
 346

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

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

350 The local server contacts the correct EPC-IS server found in the URL for  
351 the EPC in question

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

## 356 5.2.1 Serial Number Level Queries to the ONS

357 It is important to note that this version of ONS does not specify queries for what is  
358 normally considered the fully serialized SGTIN. **It specifically stops at the “Object**  
359 **Class” level.** Subsequent queries for information about a given serial number must be  
360 resolved by querying the application layer server designated by the ONS results. The  
361 ability to specify an ONS query at the serial number level of granularity as well as the  
362 architectural and economic impacts of that capability is an open issue that will be  
363 addressed in subsequent versions of this document. Its lack of mention here should not  
364 be construed as making that behavior legal or illegal.

## 365 6 ONS Nameserver Infrastructure Organization (non- 366 normative)

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

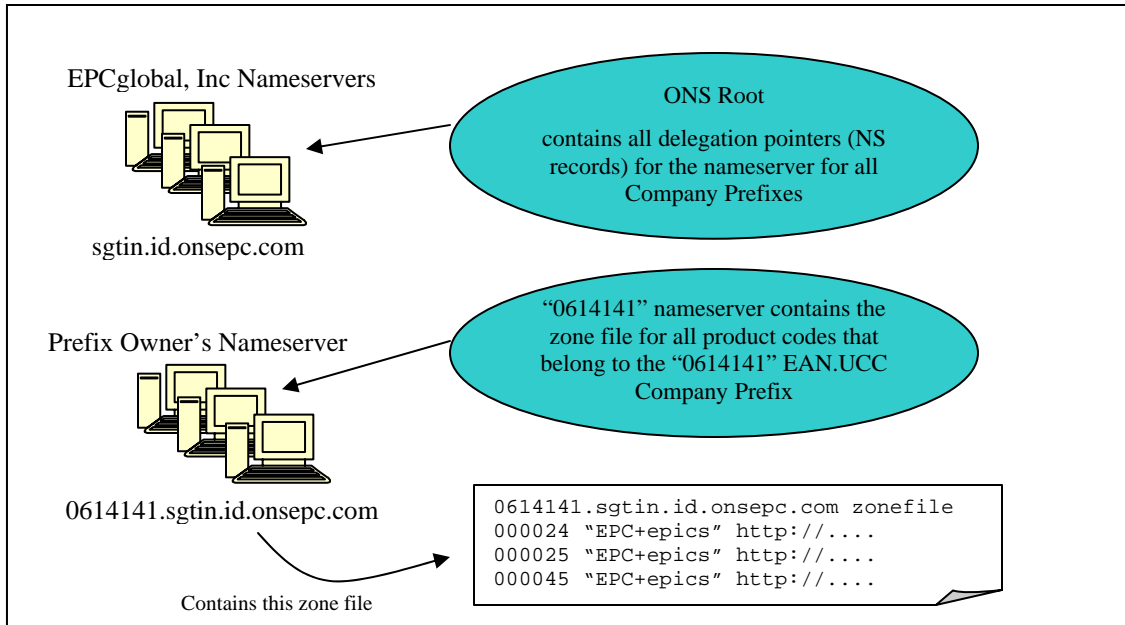
### 371 6.1 ONS Delegation Rules

372 Since ONS is specifically for looking up EPC related services it must follow the  
373 delegation rules for the various namespaces that may be part of an EPC. As mentioned  
374 in the Introduction, this specification is limited to the SGTIN namespace. Thus, while the  
375 concept of delegations discussed below is generally applicable, the specific delegation  
376 rules are specific to SGTINs.

377 An SGTIN is a serialized version of a GTIN. GTINs are part of the EAN.UCC System of  
378 product codes. In most of the namespaces that are part of this System there is the  
379 concept of an EAN.UCC Company Prefix. Company Prefixes are generally assigned by  
380 either GS1 (formerly EAN International) or GS1 US™ (formerly the Uniform Code  
381 Council, Inc.) depending on the country the registrant in question belongs to. As part of  
382 the deployment of the EPCglobal Network, GS1 and GS1 US have formed a joint  
383 venture to handle the various management duties of the network, one of which is  
384 coordinating the assignment of Company Prefixes and the provisioning of those prefixes  
385 in ONS.

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

390



391

392 Thus, in order for a company to provision their GTINs with ONS, they must request that  
 393 this be done through either GS1 US or the appropriate GS1 affiliate. The request is sent  
 394 to EPCglobal Inc™ which follows its procedures for determining the contents of the NS  
 395 record that will appear in the “sgtin.id.onsepc.com” zone. That NS record will contain the  
 396 IP addresses of the nameservers that the owner of that Company Prefix has specified.  
 397 Once these IP addresses are available in the “sgtin.id.onsepc.com” zone queries will be  
 398 routed to the Company Prefix owner’s nameservers and at that point they can determine  
 399 which Item Reference codes to publish information about.

400 Other EAN.UCC namespaces may have different rules so the above should not be  
 401 applied to namespaces such as SSCCs or GLNs until the Software Action Group  
 402 publishes the appropriate standards for those namespaces.

## 403 6.2 Zone Maintenance Guidelines

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

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

## 417 7 ONS Formal Specification

418 The formal specification of ONS is a set of procedures and rules to be followed by ONS  
 419 Clients and ONS Publishers. An ONS Client is an application that wishes to use ONS to

420 identify a service that may provide information about a specific EPC. An ONS Publisher  
421 is an entity responsible for making services available to ONS Clients by creating service  
422 pointer entries in an ONS Server. An ONS Server is an implementation of a DNS Server;  
423 because ONS differs from DNS only in what data is provided by the server, not in the  
424 operation of the server itself, there is no separate specification for an ONS Server. Any  
425 DNS server compliant with [DNS] and [RFC 3403] may be used as an ONS Server.

426 The ONS specification consists of three ingredients:

- 427 • A procedure (Section 8) that an ONS Client MUST follow in order to present a query  
428 to ONS. This procedure specifies how an EPC is converted to a DNS NAPTR query.
- 429 • A set of rules (Section 9) that ONS Publishers MUST follow to represent ONS  
430 information (namely, pointers to services for EPCs) as DNS NAPTR records within  
431 an ONS Server.
- 432 • A procedure (Section 10) that an ONS Client MUST follow in order to interpret the  
433 results of an ONS query. This procedure specifies how an ONS Client can locate a  
434 service using the information provided by the ONS Server.

## 435 **8 DNS Query Format**

436 The following specifies the procedure an ONS client MUST follow to present a query to  
437 ONS.

- 438 • Begin with an EPC represented in the pure identity URI form as  
439 defined in Section 4.1 of the EPCglobal Tag Data Standards [\[TAG](#)  
440 [Data Standards\]](#). URIs in this form are ASCII strings beginning with  
441 `urn:epc:id:`, for example,  
442 `urn:epc:id:sgtin:0614141.000024.400.`

443 Follow the procedure below to convert the URI into a domain name  
444 ending with `.onsepc.com`. For example,  
445 `000024.0614141.sgtin.id.onsepc.com`.

446 Use a DNS resolver to query for DNS Type Code 35 (NAPTR) records for  
447 the domain name from Step 2. The method for obtaining and using  
448 the DNS resolver is outside the scope of this specification. It is  
449 anticipated that this will be addressed through a companion  
450 specification that specifies a standard ONS Application Programming  
451 Interface (API). Even when a standard ONS API exists, however, an  
452 ONS Client MAY use any DNS resolver conforming to [\[DNS\]](#), using  
453 whatever API is available.

454 Follow the procedure in Section 10 to interpret the results from Step 3.

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

- 457 • Begin with an EPC represented in the pure identity URI form as  
458 defined in Section 4.3.3 of the EPCglobal Tag Data Standard [\[TAG](#)  
459 [Data Standard\]](#). For example,  
460 `urn:epc:id:sgtin:0614141.000024.400.`

461 Remove the `urn:epc:` prefix (in the example, leaving  
462 `id:sgtin:0614141.000024.400`).

463 Remove the serial number field. In all tag formats currently defined in  
 464 [\[TAG Data Standard\]](#) (SGTIN, SSCC, SGLN, GRAI, GIAI, and GID),  
 465 the serial number field is the rightmost period (.) character and all  
 466 characters to the right of it. (In the example, this leaves  
 467 `id:sgtin:0614141.000024`)

468 Replace each colon (:) character with a period (.) character (in the  
 469 example, leaving `id.sgtin.0614141.000024`)

470 Invert the order of the remaining period-delimited fields (in the example,  
 471 leaving `000024.0614141.sgtin.id`)

472 Append `.onsepc.com`. In the example, the result is  
 473 `000024.0614141.sgtin.id.onsepc.com`.

## 474 9 DNS Records for ONS

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

478 The contents of a DNS NAPTR record are logically formatted as follows:  
 479

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

480

481 ONS Publishers MUST obey the following rules:

- 482 • The **Order** field MUST be zero.

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

- 488 • The **Pref** field MUST be a non-negative integer. The value of the Pref field is an  
 489 ordinal that specifies that the service in one record is preferred to the service in  
 490 another record having the same Service field. An ONS Client SHOULD attempt  
 491 to use a service having a lower Pref number before using an equivalent service  
 492 having a higher Pref number.
- 493 • The **Flags** field MUST be set to 'u', indicating that the **Regexp** field contains a URI.
- 494 • The **Service** field contains an indicator of the type of service that can be found at the  
 495 URI in question. This feature allows for the ONS service to indicate different service  
 496 end points for different classes of service. The value of the Service field MUST  
 497 consist of the string EPC+ followed by the name of a service registered with the  
 498 EPCglobal Network Protocol Parameter Registry. The Service field identifies what  
 499 type of service is accessible through the URL provided by the Regexp field of this  
 500 record. See Section 10 for examples of service types, and Section 15 for a list of  
 501 services that make up an initial set of registered service classes.

- 502 • The **Replacement** field is not used by the EPCglobal Network but since it is a  
503 special DNS field its value is set to a single period ('.') instead of simply a blank.
- 504 • The **Regexp** field specifies a URL for the service being described. The value of this  
505 field MUST be the string !^.\*\$! (the six character sequence consisting of an  
506 exclamation point, a caret, a period, an asterisk, a dollar sign, and another  
507 exclamation point), followed by a URL, followed by an exclamation point (!)  
508 character.

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

519 *The general form of the Regexp field is as a Posix Extended Regular Expression. This*  
520 *form states that the first character encountered is the field delimiter between the regular*  
521 *expression and the replacement portion of the entire rewrite expression. In the above*  
522 *example the delimiter is the exclamation point (!) character. The regular expression*  
523 *portion is ^.\*\$ which equates to 'match anything'. The replacement portion is*  
524 *http://example.com/cgi-bin/epcis. The choice of '!' as the delimiter*  
525 *instead of a more traditional '/' makes the entire line much easier to read and less error*  
526 *prone.*

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

## 530 **10 Processing ONS Query Responses**

531 ONS Clients MUST use the following procedure to interpret the results returned by an  
532 ONS query as formulated in Section 8.

- 533 • The result from the ONS query is a set of NAPTR records as  
534 described in Section 9.
- 535 From among the results from Step 1, select those records whose Service  
536 field names the desired service. If there is no such record, stop: a  
537 pointer to the desired service is not available from ONS for the  
538 specified EPC.
- 539 From among the results from Step 2, select those records having the  
540 lowest value in the Pref field.
- 541 From among the results from Step 3, select a record at random.
- 542 Extract the service URL from the record from Step 4, by extracting the  
543 substring between the initial !^.\*\$! and the final ! character.
- 544 Attempt to use the service URL from Step 5.

545 If Step 6 is not successful, go back to Step 4, using a different record  
 546 from among the records from Step 3. If all records from Step 3 have  
 547 been tried, go back to Step 3 using records from Step 2 having the  
 548 next lowest value in the Pref field. If all records from Step 2 have  
 549 been tried, stop: no service is available.

## 550 11 Examples (non-normative)

551 In the following examples the EPC in question is  
 552 urn:epc:id:sgtin:0614141.011015.583865 which represents an Example Corporation  
 553 Model 100 Widget.

554 The ONS Client application attempts to learn about this product by first following the  
 555 procedure in Section 8, which converts the EPC into a domain-name:  
 556 011015.0614141.sgtin.id.onsepc.com

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

Order	Pref	Flags	Service	Regexp	Replacement
0	0	u	EPC+ws	!^\.*\$!http://example.com/autoid/widget100.wsd!	.
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!	.

559

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

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

### 566 11.1 Finding a WSDL file for a product

567 One of the simplest but most powerful examples is where an ERP application is Web  
 568 Services [\[Web Services\]](#) enabled. This particular application is capable of learning about  
 569 new products by making various application specific web services calls to public  
 570 interfaces made available by the manufacturer. In this case the application can simply  
 571 use the ONS to look to see if a WSDL file exists that describes the Web Services it  
 572 requires.

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

## 579 **11.2 Finding an authoritative EPCIS server for a product**

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

## 584 **11.3 Finding an HTML formatted web page description of a** 585 **product**

586 This example shows how a very lightweight service can be deployed almost immediately  
587 using nothing more than an existing externally available corporate web server containing  
588 existing product content. By inserting the third record in the results list above, Example  
589 Corp can easily point applications to authoritative product data without any modifications  
590 to their systems. Applications that understand this service can be very lightweight, using  
591 existing web browsers to display the content.

## 592 **11.4 Finding an XML-RPC gateway to the Web Service interfaces**

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

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

602

## 603 **12 References**

### 604 **Willinger and Doyle**

605 Willinger, W. and Doyle, John. *Robustness and the Internet: Design and*  
606 *evolution*, 2002. (See [http://netlab.caltech.edu/pub/papers/part1\\_vers4.pdf](http://netlab.caltech.edu/pub/papers/part1_vers4.pdf))

### 607 **EPC**

608 Brock, David. *The Electronic Product Code (EPC): A Naming Scheme for*  
609 *Physical Objects*, 2001. (See [http://www.autoidlabs.org/whitepapers/MIT-](http://www.autoidlabs.org/whitepapers/MIT-AUTOID-WH-002.pdf)  
610 [AUTOID-WH-002.pdf](http://www.autoidlabs.org/whitepapers/MIT-AUTOID-WH-002.pdf))

### 611 **RFC 2826**

612 (Internet Architecture Board). *RFC 2826: IAB Technical Comment on the Unique*  
613 *DNS Root*, 2000. (See <http://www.ietf.org/rfc/rfc2826.txt>)

### 614 **DNS**

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

### 617 **Web Services**

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

- 620 **XML-RPC**
- 621 Winer, Dave. *XML-RPC Specification*, 1999. (See <http://www.xml-rpc.com/spec>)
- 622 **RFC 2396**
- 623 T. Berners-Lee, R. Fielding, L. Masinter. *Uniform Resource Identifiers (URI):*
- 624 *Generic Syntax*, 1998. (See <http://www.ietf.org/rfc/rfc2396.txt>)
- 625 **RFC 3401**
- 626 Mealling, Michael. *Dynamic Delegation Discovery System (DDDS) Part One: The*
- 627 *Comprehensive DDDS*, 2002. (See <http://www.ietf.org/rfc/rfc3401.txt>)
- 628
- 629
- 630 **RFC 3402**
- 631 Mealling, Michael. *Dynamic Delegation Discovery System (DDDS) Part Two: The*
- 632 *Algorithm*, 2002. (See <http://www.ietf.org/rfc/rfc3402.txt>)
- 633 **RFC 3403**
- 634 Mealling, Michael. *Dynamic Delegation Discovery System (DDDS) Part Three:*
- 635 *The Domain Name System (DNS) Database*, 2002. (See
- 636 <http://www.ietf.org/rfc/rfc3403.txt>)
- 637 **Registry**
- 638 Mealling, Michael. *EPC Network Protocol Parameter Registry*, 2002. (See
- 639 <http://www.ietf.org/rfc/rfc3403.txt>)
- 640 **TAG Data Standards**
- 641 EPCglobal, "EPC Tag Data Standards Version 1.1 Rev.1.24," EPCglobal
- 642 Standard Specification, April 2004,
- 643 [http://www.epcglobalinc.org/standards\\_technology/EPCTagDataSpecification11r](http://www.epcglobalinc.org/standards_technology/EPCTagDataSpecification11r)
- 644 [ev124.pdf](http://www.epcglobalinc.org/standards_technology/EPCTagDataSpecification11rev124.pdf).
- 645

## 646 **13 Appendix A – Glossary (non-normative)**

### 647 **Auto-ID**

648 "Automatic Identification" -- An open, global network that can identify anything,

649 anywhere, automatically.

### 650 **Domain-name**

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

652 Internet

### 653 **DNS**

654 See Domain Name Service

### 655 **Domain Name Service**

656 An infrastructure level Internet service used to discover information about a

657 domain name. It was originally developed to map a host name to an IP address,

658 but has since been extended to other uses (such as ENUM, which maps a phone

659 number to one or more communication services.

- 660 **EPC**
- 661 See Electronic Product Code
- 662 **EPCIS**
- 663 EPC Information Service – A series of EPC Network specific standards defining  
664 various methods of data exchange and metadata lookup.
- 665 **Electronic Product Code**
- 666 An abstract namespace made up of an EPC Manager Number, an Object Class  
667 code, and a Serial Number, or a subset thereof.
- 668 **EPC Manager Number**
- 669 A code that identifies a manufacturer of objects
- 670 **ONS**
- 671 See Object Name Service
- 672 **ONS Root**
- 673 The domain-name that is appended to the manager id and which acts as the ‘top’  
674 of the DNS tree that contains all of the EPC entries. This root domain is  
675 “onsepc.com”
- 676 **Object Name Service**
- 677 A resolution system, based on DNS, for discovering authoritative information  
678 about an EPC
- 679 **Object Class code**
- 680 A code that identifies a particular type of object that is created by a particular  
681 manufacturer
- 682 **NAPTR**
- 683 "Naming Authority PoinTeR" -- A DNS record type (35) that contains information  
684 about a specific delegation point within some other namespace using regular  
685 expressions.
- 686 **PML**
- 687 "Physical Markup Language" -- generally information obtained from a Reader or  
688 similar sensor
- 689 **Reader**
- 690 A radio enabled device that communicates with a tag
- 691 **RFID**
- 692 "Radio Frequency Identification" -- A method of identifying unique items using  
693 radio waves. The big advantage over bar code technology is lasers must see a  
694 bar code to read it. Radio waves do not require line of sight and can pass  
695 through materials such as cardboard and plastic.
- 696 **Regular Expression**
- 697 A standard language for pattern matching within a string of characters and for  
698 composing new strings based on matched subcomponents of the original string  
699 (i.e. a search and replace function)

700 **Serial Number**

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

702 **tag**

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

705 **URI**

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

709 **URL**

710 "Uniform Resource Locator" -- A deprecated term usually used to denote the  
711 subclass of URIs that contain DNS domain-names in their authority section.

712 **14 Appendix B -- DDDS Application Specification (non-**  
713 **normative)**

714 The use of NAPTR records is governed by a series of RFCs that define something called  
715 the Dynamic Delegation Discovery Service. RFC 3401 [\[RFC 3401\]](#) is the first in the  
716 series and provides an introduction to the series. In order to safely use NAPTR records  
717 on the public network a specification must exist that describes the values of the various  
718 fields. This appendix contains that specification which, when approved by the SAG  
719 process, will be extracted and published as an RFC itself.

720 **14.1 Application Unique String**

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

722 **14.2 First Well Known Rule**

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

727 **14.3 Expected Output**

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

731 **14.4 Valid Databases**

732 At present only one DDDS Database is specified for this Application. RFC 3403 [\[RFC](#)  
733 [3403\]](#) specifies a DDDS Database that uses the NAPTR DNS resource record to contain  
734 the rewrite rules. The Keys for this database are encoded as domain-names. The  
735 conversion method for this database is as follows:

- 736 1. Remove the 'urn:epc:' header
- 737 2. Remove the serial number field
- 738 3. Invert the order of the remaining fields

- 739                   4. Convert all ‘.’ characters to ‘.’  
740                   5. Append “.onsepc.com”

## 741 **14.5 Valid Flags**

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

## 744 **14.6 Service Parameters**

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

747                   service\_field = "EPC+" service\_name

748                   service\_name = ALPHA \*31ALPHANUM

749 The valid values for 'service\_name' and the process for registering new services are  
750 found in the ONS Service Registry discussed in Appendix C. It is important to note that  
751 the “+” character is not allowed in a service\_name. This allows for future expansion of  
752 the service field if it becomes apparent that service\_names may need parameters.

## 753 **15 Appendix C -- Service Field Registrations (non- 754 normative)**

755 The 'service\_name' portion of the Service field is a managed space where the values  
756 that are available for use are found in the [EPCglobal Network Protocol Parameters  
757 Registry](#). The goal is to balance the ability to innovate with new services with the desire  
758 for interoperability between services. If the service\_name simply ends up denoting non-  
759 interoperable, proprietary services then the total value of the system is significantly  
760 reduced. Conversely, limiting all services to simply those that are standardized by the  
761 EPCglobal SAG standards process limits the ability of the system to innovate.

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

768 The Registry entry for ONS Service types will be created with the /ons/service\_name/  
769 pathname. Files in that directory will be named after Service Name field in the template  
770 followed by the '.txt' file extension. The contents of the file will be the template supplied  
771 for the registration. For example, the 'epcis' template below would be found at  
772 [http://onsepc.com/ons/service\\_name/epcis.txt](http://onsepc.com/ons/service_name/epcis.txt) and would contain the just the fields  
773 mentioned below. Registrations are First Come First Served. Registration requires a  
774 published Recommendation from EPCglobal.

### 775 **15.1 Registration Template**

776 A document specifying a service\_name must contain the following template. This  
777 template is then entered into the registry as soon as the document is published.

#### 778 **Service Name:**

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

780 **Functional Specification:**

781 A pointer to publicly available documentation for the service.

782 **Valid URI schemes:**

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

785 **Security Considerations:**

786 Any security issues associated with use of this service

787 **15.2 Service Registrations**

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

790 • **Service Name: epcis**

791 **Functional Specification:** The EPC Information Service Specification

792 **Valid URI schemes:** http, https

793 **Security Considerations:** See the Security Considerations section of the EPC  
794 Information Service Specification

795 • **Service Name: ws**

796 **Functional Specification:** A generic Web Services [\[Web Services\]](#) based service  
797 where the application must negotiate what services are available by investigating the  
798 WSDL file found at the URI in the Regexp

799 **Valid URI schemes:** http, https

800 **Security Considerations:** Web Services utilize a great deal of existing Internet  
801 infrastructure and protocols. It is very easy to use some of them in insecure ways.  
802 Any usage of Web Services should be done in the context of a thorough  
803 understanding of the dependencies, especially as it relates to the DNS and HTTP.

804 • **Service Name: html**

805 **Functional Specification:** Simply returns a URI that will resolve to an HTML page  
806 on some server. The assumption is that this page contains information about the  
807 product in question.

808 **Valid URI schemes:** http, https, ftp

809 **Security Considerations:** None not already inherent in the use of the  
810 WorldWideWeb.

811 • **Service Name: xmlrpc**

812 **Functional Specification:** A URI that denotes an HTTP POST capable service on  
813 some server that is expecting XML-RPC [\[XML-RPC\]](#) compliant connection.

814 **Valid URI schemes:** http, https

815 **Security Considerations:** None not already inherent in the use of the  
816 WorldWideWeb.

817