Object Naming Service (ONS)
Version 1.0

EPCglobal Ratified Specification
Version of October 4, 2005

Abstract

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

Status of this document

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

Comments on this document should be sent to the EPCglobal Software Action Group (SAG) mailing list (sag_ons@epclinklist.epcglobalinc.org).

Table of contents

1 Introduction

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

2 Terminology and Typographical Conventions

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

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

The following typographical conventions are used throughout the document:

- ALL CAPS type is used for the special terms from [ISODir2] enumerated above.
- `Monospace` type is used to denote programming language, UML, and XML identifiers, as well as for the text of XML documents.

➢ Placeholders for changes that need to be made to this document prior to its reaching the final stage of approved EPCglobal specification are prefixed by a rightward-facing arrowhead, as this paragraph is.

3 Background Information (non-normative)

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

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

4 EPCglobal Network Architecture (non-normative)

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

4.1 Electronic Product Code (EPC)

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

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

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

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

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

4.2 EPCglobal Network Software Architecture Components

The EPCglobal Network Architecture as in Fig. 1 shows the high-level components of the EPCglobal Network. This section highlights and makes reference to The EPCglobal Network Architecture Framework Version 1, July 2005. Available on the EPCglobal website at:
http://www.epcglobalinc.org/standards_technology/Final-epcglobal-arch-20050701.pdf

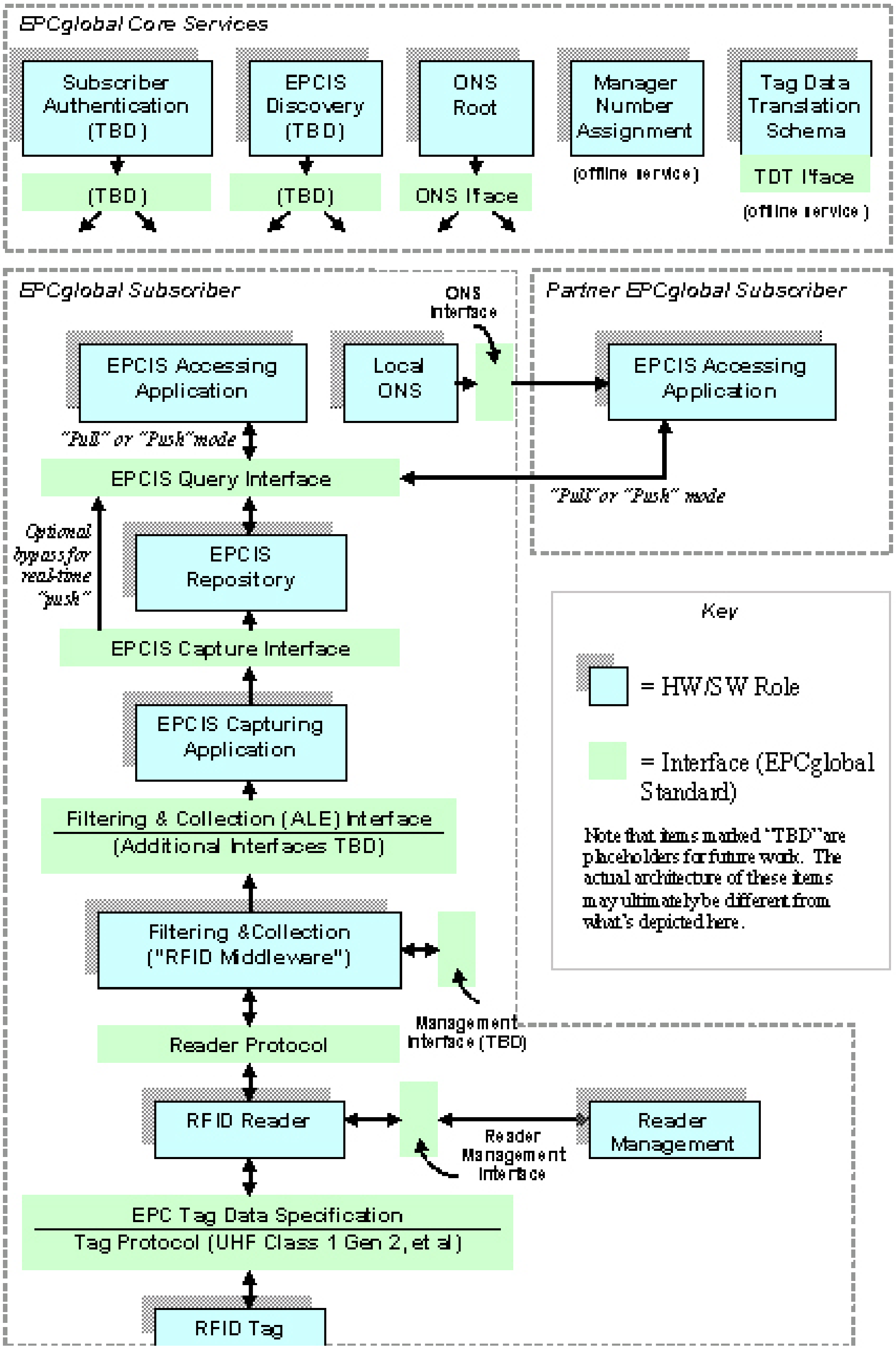

EPCglobal Core Services
Subscriber Authentication (TBD)
EPCIS Discovery (TBD)
ONS Root
Manager Number Assignment
Tag Data Translation Schema
(TBD)
(TBD)
ONS I'face
(offline service)
TDT I'face
(offline service)
EPCglobal Subscriber
ONS Interface
Partner EPCglobal Subscriber
EPCIS Accessing Application
Local ONS
EPCIS Accessing Application
"Pull" or "Push" mode
EPCIS Query Interface
"Pull" or "Push" mode
Optional bypass for real-time "push"
EPCIS Repository
EPCIS Capture Interface
EPCIS Capturing Application
Filtering & Collection (ALE) Interface
(Additional Interfaces TBD)
Filtering & Collection ("RFID Middleware")
Management Interface (TBD)
Reader Protocol
RFID Reader
Reader Management Interface
Reader Management
EPC Tag Data Specification
Tag Protocol (UHF Class 1 Gen 2, et al)
RFID Tag
Key
= HW/SW Role
= Interface (EPCglobal Standard)
Note that items marked "TBD" are placeholders for future work. The actual architecture of these items may ultimately be different from what's depicted here.

Figure 1: EPCglobal Network Architecture: Components and Layers

Components in blue are “roles”, not discrete pieces of software. Components in green are interfaces used between two roles. The yellow components labeled “ERP” correspond to locations where one or more ERP systems might provide or interface with a particular role. These components from the figures above are described in the sections below:

- *Readers*
  Make multiple observations of RFID tags while they are in the read zone.
- *Reader Protocol Interface*
  Delivers raw tag reads from Readers to Middleware. Events at this interface say “Reader A saw EPC X at time T.”
- *Middleware*
  Accumulates and filters raw tag reads
- *ALE Interface*
  Delivers consolidated, filtered tag read data from Middleware to Local Application. Events at this interface are “At Location L, between time T1 and T2, the following EPCs were observed” where the list of EPCs has no duplicates and has been filtered by appropriate criteria.
- *EPC Capturing Application*
  Recognizes the occurrence of EPC-related business events, and delivers these as EPCIS data
- *EPCIS Capture Interface*
  Provides a path for communicating EPCIS events generated by EPCIS Capturing Applications to other roles that require them, including EPCIS Repositories, internal EPCIS Accessing Applications, and Partner EPCIS Accessing Applications.
- *EPCIS Repository*
  Records EPCIS-level events generated by one or more EPCIS Capturing Applications, and makes them available for later query by EPCIS Accessing Applications.
- *EPCIS Query Interface*
  Provides means whereby an EPCIS Accessing Application can request EPCIS data from an EPCIS Repository or an EPCIS Capturing Application, and the means by which the result is returned.
- *EPCIS-Accessing Application*
  Software that carries out overall enterprise business processes, such as warehouse management, shipping and receiving, historical throughput analysis, and so forth, aided by EPC-related data.
- *Local ONS*
  Fulfills ONS lookup requests for EPCs within the control of the enterprise that operates the Local ONS; that is, EPCs for which the enterprise is the EPC Manager.
- *EPCIS Accessing Application*
  An EPCIS-enabled Application of a trading partner. Partner Applications may be granted access to a subset of the information that is available within the enterprise.
- *Tag Data Translation Schema*
  Provides a machine-readable file that defines how to translate between EPC

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

- *Manager Number Assignment*
  Ensures global uniqueness of EPCs by maintaining uniqueness of EPC Manager Numbers assigned to EPCglobal Subscribers

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

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

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

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

5 ONS Introduction

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

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

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

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

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

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

5.1.1 Client’s View

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

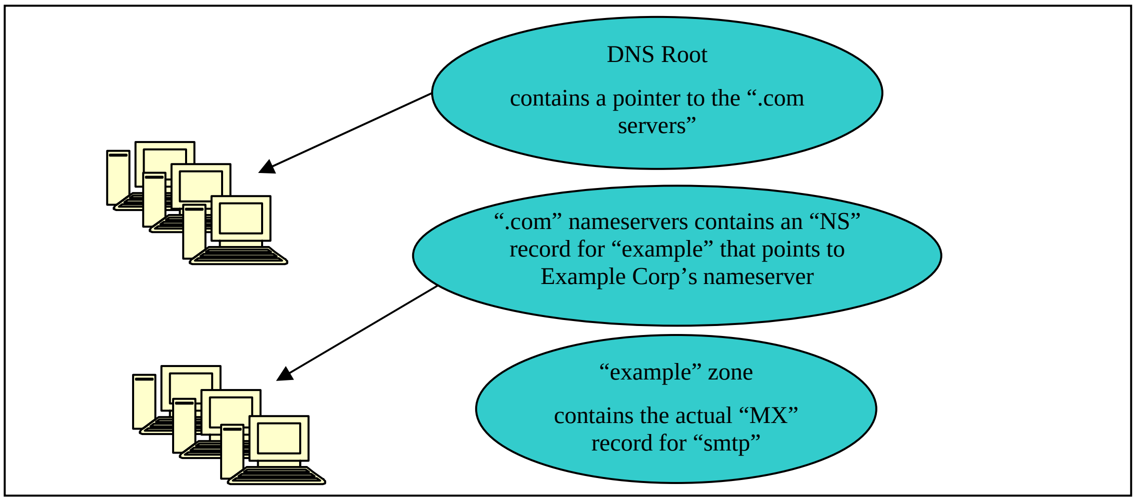
The practical consequence of this is that the DNS API that client applications will use to perform an ONS query is actually very simple. There are only three pieces of information needed: the nameserver to ask, the domain-name in question, and the type of record that is being requested. The first item,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

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

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

5.1.2 Publisher’s View

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



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

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

5.2 ONS's Usage of DNS

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

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

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

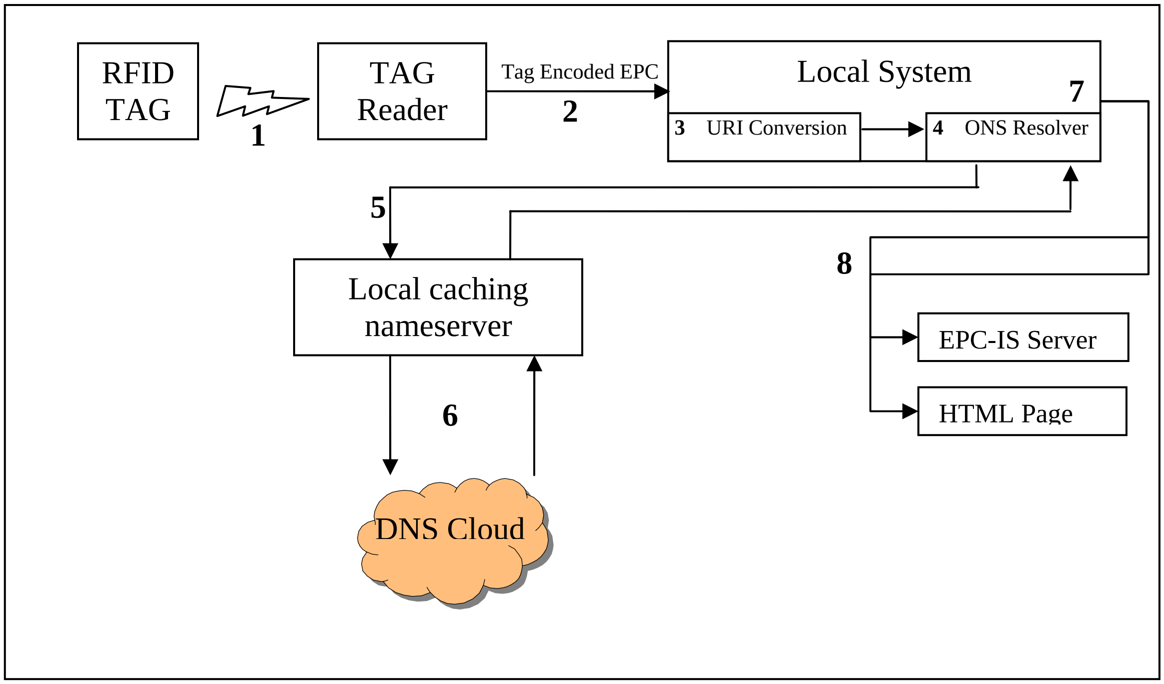


Figure 3: A typical ONS query.

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.)

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

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

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

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

5.2.1 Serial Number Level Queries to the ONS

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

6 ONS Nameserver Infrastructure Organization (non-normative)

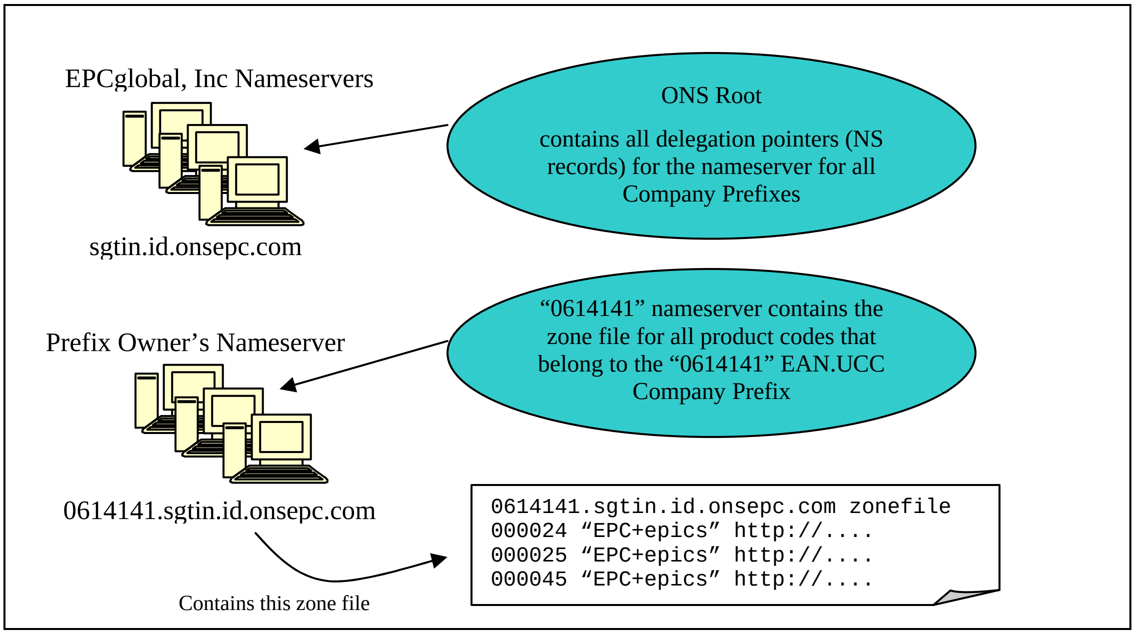
The previous section covered DNS and how ONS uses DNS from a client point of view. What it did not cover was the processes and procedures for making ONS data available in the proper form for the client. The main issues are the delegation hierarchy and zone maintenance.

6.1 ONS Delegation Rules

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

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

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



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

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

6.2 Zone Maintenance Guidelines

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

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

7 ONS Formal Specification

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

The ONS specification consists of three ingredients:

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

8 DNS Query Format

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

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

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

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

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

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

- Begin with an EPC represented in the pure identity URI form as defined in Section 4.3.3 of the EPCglobal Tag Data Standard [TAG Data Standard]. For example, `urn:epc:id:sgtin:0614141.000024.400`.

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

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

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

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

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

9 DNS Records for ONS

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

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

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

ONS Publishers MUST obey the following rules:

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

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

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

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

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

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

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

10 Processing ONS Query Responses

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

- The result from the ONS query is a set of NAPTR records as described in Section 9.

From among the results from Step 1, select those records whose Service field names the desired service. If there is no such record, stop: a pointer to the desired service is not available from ONS for the specified EPC.

From among the results from Step 2, select those records having the lowest value in the Pref field.

From among the results from Step 3, select a record at random.

Extract the service URL from the record from Step 4, by extracting the substring between the initial `!^.*$!` and the final `!` character.

Attempt to use the service URL from Step 5.

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

11 Examples (non-normative)

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

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

The application then queries the DNS for NAPTR records for that domain name and receives 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!	.

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

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

11.1 Finding a WSDL file for a product

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

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

11.2 Finding an authoritative EPCIS server for a product

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

11.3 Finding an HTML formatted web page description of a product

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

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

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

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

12 References

Willinger and Doyle

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

EPC

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

RFC 2826

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

DNS

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

Web Services

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

XML-RPC

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

RFC 2396

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

RFC 3401

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

RFC 3402

Mealling, Michael. *Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm*, 2002. (See http://www.ietf.org/rfc/rfc3402.txt)

RFC 3403

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

Registry

Mealling, Michael. *EPC Network Protocol Parameter Registry*, 2002. (See http://www.ietf.org/rfc/rfc3403.txt)

TAG Data Standards

EPCglobal, “EPC Tag Data Standards Version 1.1 Rev.1.24,” EPCglobal Standard Specification, April 2004, http://www.epcglobalinc.org/standards_technology/EPCTagDataSpecification11rev124.pdf.

13 Appendix A – Glossary (non-normative)

Auto-ID

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

Domain-name

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

DNS

See Domain Name Service

Domain Name Service

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

EPC

See Electronic Product Code

EPCIS

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

Electronic Product Code

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

EPC Manager Number

A code that identifies a manufacturer of objects

ONS

See Object Name Service

ONS Root

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

Object Name Service

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

Object Class code

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

NAPTR

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

PML

"Physical Markup Language" -- generally information obtained from a Reader or similar sensor

Reader

A radio enabled device that communicates with a tag

RFID

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

Regular Expression

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

Serial Number

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

tag

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

URI

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

URL

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

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

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

14.1 Application Unique String

The Application Unique String is the EPC in URI form.

14.2 First Well Known Rule

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

14.3 Expected Output

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

14.4 Valid Databases

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

1. Remove the '`urn:epc:`' header
2. Remove the serial number field
3. Invert the order of the remaining fields

4. Convert all ‘:’ characters to ‘.’
5. Append “.onsepc.com”

14.5 Valid Flags

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

14.6 Service Parameters

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

service_field = "EPC+" service_name

service_name = ALPHA *31ALPHANUM

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

15 Appendix C -- Service Field Registrations (non-normative)

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

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

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

15.1 Registration Template

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

Service Name:

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

Functional Specification:

A pointer to publicly available documentation for the service.

Valid URI schemes:

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

Security Considerations:

Any security issues associated with use of this service

15.2 Service Registrations

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

- **Service Name: epcis**

  Functional Specification: The EPC Information Service Specification
  Valid URI schemes: http, https
  Security Considerations: See the Security Considerations section of the EPC Information Service Specification

- **Service Name: ws**

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

- **Service Name: html**

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

- **Service Name: xmlrpc**

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

