Published February 1, 2002. Distribution restricted to Sponsors until May 1, 2002.

TECHNICAL MANUAL

The Object Name Service

Version 0.5 (Beta)

Oat Systems & MIT Auto-ID Center

AUTO-ID CENTER MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 77 MASSACHUSETTS AVENUE, BLDG 3-449, CAMBRIDGE, MA 02139-4307, USA

ABSTRACT

This document gives a technical description of the Object Name Service Framework. It should be read by IT professionals interested in deploying and maintaining the Object Name Service. The reader is expected to understand the basic concepts of operating systems, networking, programming and scripting.

Published February 1, 2002. Distribution restricted to Sponsors until May 1, 2002.

TECHNICAL MANUAL

The Object Name Service

Version 0.5 (Beta)

Contents

TECHNICAL MANUAL

The Object Name Service

Version 0.5 (Beta)

Contents

AUDIENCE

This document gives a technical description of the Object Name Service Framework. It should be read by IT professionals interested in deploying and maintaining the Object Name Service. The reader is expected to understand the basic concepts of operating systems, networking, programming and scripting.

1. INTRODUCTION

This document describes an **Object Name Service (ONS)** framework to locate networked services for tagged objects. Specifically, networked services for an object can be identified based on the unique **electronic product code (EPC)** tagged to that object.

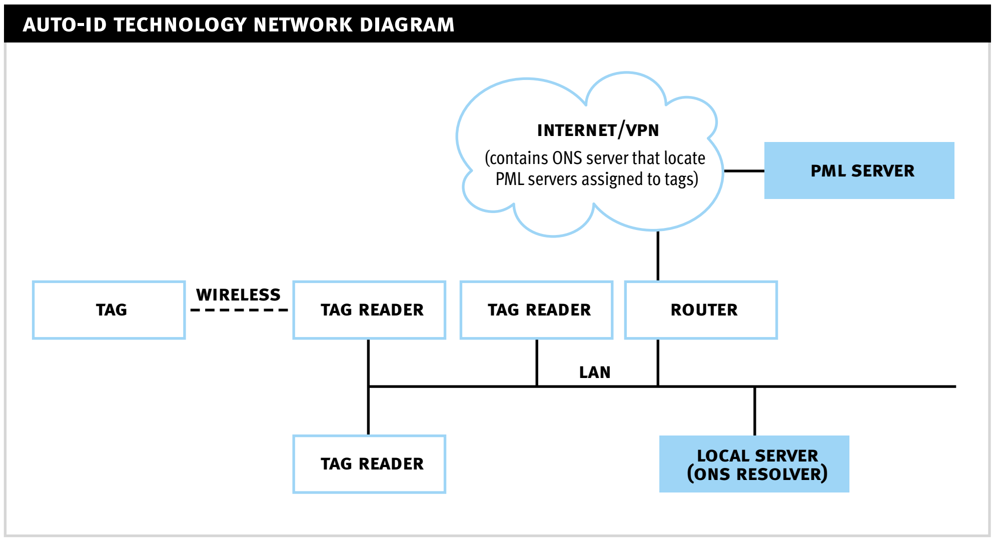
A **reader** identifies the EPC in a tag, preferably without manual intervention. For example, a radio-frequency sensor can detect all RFIDs in a certain detection range around it.

Based on its tag EPC, an object can be associated with networked services. A **network service** is a remote service on the Internet or a Virtual Private Network (VPN) offered to provide and store information regarding that object. A typical network service could offer product information regarding the object. The ONS framework helps readers, or processing units integrated with the readers locate these services.

Currently, the ONS service is used to locate the **PML servers** associated with an EPC. A PML server is a simple webserver that serves information about that object in the **Physical Markup Language**.

Figure 1 describes a typical location implementing Auto-ID technology. The ONS resolver implemented in the location's local server helps the server obtain global information about the EPCs detected using tag readers.

Figure 1: Network diagram for a typical location implementing Auto-ID technology


AUTO-ID TECHNOLOGY NETWORK DIAGRAM
INTERNET/VPN
(contains ONS server that locate PML servers assigned to tags)
PML SERVER
TAG
WIRELESS
TAG READER
TAG READER
ROUTER
LAN
TAG READER
LOCAL SERVER (ONS RESOLVER)

Section 2 describes the requirements of the ONS framework. Section 3 proposes an ONS framework built over the existing DNS framework. In Section 4, we look at the implementation details of ONS packages. Finally, Section 5 contains some concluding remarks.

2. REQUIREMENTS

This section describes the various requirements of the ONS framework. We will see why the ONS framework should be hierachical, cache-friendly, redundant, and extendible.

The ONS framework deals with the management and lookup of **mapping information**. Mapping information can be used to find out the PML servers associated with an EPC.

The EPC namespace is hierarchically assigned for ease in management. An EPC is of the form
<version number> . <manufacturer ID> . <product ID> . <serial number>,

where
– **version number** is a constant that specifies the format of EPC. The version is 2 bits long if the EPC does not begin with the bits 00. Otherwise, the version is 8 bits long.
– **manufacturer ID** is a bit string identifying the manufacturer
– **product ID** is a bit string identifying the product provided by the manufacturer
– **serial number** is a bit string identifying the serial number of the product provided by the manufacturer

Each version defines the bit-length of the manufacturer ID, product ID and serial number.

The **EPC namespace authority**[1] will assign manufacturers with unique manufacturer IDs. The manufacturers can then independently assign unique product IDs to their products. The manufacturers or product-specific divisions of the manufacturers can then assign unique serial numbers to various instances of a product.

[1] The EPC namespace authority could be the Auto-ID center or any other standards organization chosen by the Auto-ID Center

The mapping information in the ONS framework should therefore be maintained in a hierarchical fashion involving the EPC namespace authority, the manufacturers and product-specific divisions of the manufacturers. Thus we have:

Requirement 1. The ONS framework should allow hierarchical management of the mapping information.

Every access to any information about an EPC involves an **ONS lookup**. Without optimization, an ONS lookup will be performed every time a PML server is accessed. This is a considerable overhead due to the hierarchical organization of the mapping information. Thus we have:

Requirement 2. The ONS framework should allow the caching of mapping information held in ONS servers.

Since the ONS and PML infrastructure is at the heart of the Auto-ID object management architecture, we have:

Requirement 3. The ONS framework should allow the same mapping information to be stored in redundant ONS servers.

Requirement 4. The ONS framework should allow EPCs to be mapped to redundant PML servers.

The size of the EPC can vary depending on the version number of the EPC. New versions holding larger EPCs may be added as tag technology improves. Thus we have:

Requirement 5. The ONS framework should allow the addition of new versions of EPCs with no changes to its software and hardware components.

The above five requirements drive the design of our ONS framework. In the next section, we will look at the architecture of the ONS framework.

3. SYSTEM ARCHITECTURE

This section gives an overview of the ONS architecture. The ONS architecture is a distributed framework consisting of:

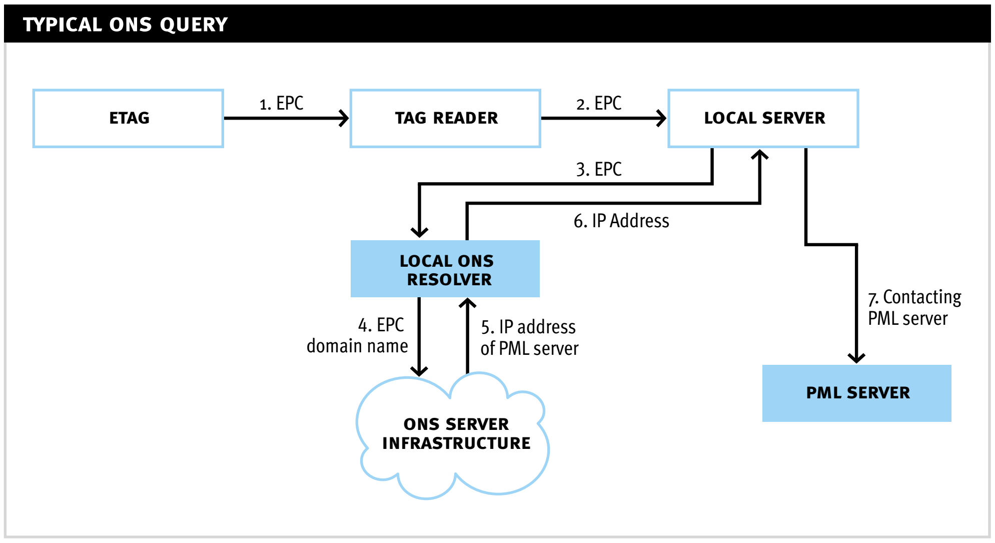
1. Mapping information
This information is hierarchically distributed across ONS servers for manageability. Mapping information is stored in records that express the mapping between a range of EPCs and a PML server.

2. ONS servers
These are servers that respond to queries requesting the IP addresses of PML servers associated with an EPC. Each server has **authoritative** mapping information regarding certain EPCs, and **cached** mapping information regarding other EPCs.

3. ONS resolver
A resolver submits queries to ONS servers to obtain the network location of PML servers.

Figure 2 shows the steps involved in a typical ONS query. As shown in the figure, a local ONS server, uses the ONS infrastructure to locate the IP address of the PML server associated with an EPC.

Figure 2: A typical ONS query



Section 3.1 describes the existing DNS (Domain Name Service) architecture, and elaborates on how the DNS framework matches most of the requirements for the ONS framework. In Section 3.2, we will look at how **translation format strings** can be used to implement ONS over DNS. Section 3.3 gives an algorithm for translating EPCs to domain names.

3.1. Overview of the DNS Framework

The Domain Name Service (DNS) framework is central to the Internet. DNS is designed to provide a consistent hierarchical name space called **domain name**, based on which a remote service can be accessed. The DNS framework is robust, distributed, tuned for performance, interoperable and universally accepted.

For example, when the domain name `www.mit.edu` is accessed in a browser, the browser picks any one of the IP addresses associated with that domain name, and performs a http query to that machine.

A domain name consists of a sequence of **labels**. For example the domain name `www.mit.edu` consists of the label sequence `www,` `mit,` and `edu.` A label consists of alphanumeric characters and dashes. The `'.'` character separates labels in a domain name.

A domain is said to be the **suffix** of another domain if the latter domain label sequence ends with the former's label sequence. For example, `edu` is a domain-suffix of `www.mit.edu`

The DNS architecture consists of DNS servers holding authoritative mapping information. Each entry in the mapping, called a **DNS resource record**. Common DNS records are:

– **A records:** These address records point a domain name to an IP address. For example, the DNS record `www.mit.edu A 18.58.3.83` points the domain name `www.mit.edu` to the IP address `18.58.3.83.` Multiple A records for the same domain name indicate that any of the given machines can be accessed for services in that domain name.

– **NS records:** These name-server records point a domain-suffix to an authoritative DNS server that is closer to the given domain. For example, the entry `edu NS ns.edu,` directs the request for `www.mit.edu` to the DNS name server `ns.edu`. Looking at this record, a DNS resolver searching for the IP address of `www.mit.edu` will query the DNS server at `ns.edu` to resolve at the `.edu` namespace. The resolver iterates this process until it reaches the DNS server holding the A record for `www.mit.edu`.

  The DNS server providing this record will also provide an additional record with the IP addresses associated with `ns.edu`. For example, an address record, say `ns.edu NS 18.58.1.171` could be provided along with the NS record.

– **TXT records:** These text records point a domain name to some textual information about that domain. For example, `www.mit.edu TXT "hi there"` provides some additional textual information about `www.mit.edu`

A set of root DNS servers are authoritative for the root domain. The root domain is an empty domain consisting of no labels, represented by `'.'`.

The DNS framework satisfies Requirement 1. A DNS resolver uses DNS servers to lookup the IP addresses associated with a domain. When no DNS records are cached, every request is sent to a root server, i.e., a DNS server authoritative for the root domain. The root server points the resolver to a DNS server closer to the given domain name. The DNS resolver iterates this process to obtain the addresses of the DNS servers authoritative for the given domain name.

The above method is inefficient because more than one DNS query is made to lookup a domain. However, the DNS framework allows **record caching** to make DNS lookups more efficient for the common case.

To enable caching, every DNS record stored in an authoritative DNS server is associated with a **time-to-live** (TTL). The TTL is the time for which the record can be cached. DNS servers that are not authoritative for a given records can cache the record for the period for which the TTL has not expired. Furthermore, a non-authoritative DNS server can serve this record to another DNS server requesting this record after reducing the initial TTL by the time for which the record was cached. In this manner, DNS servers can co-operate with each other in distributing the load of DNS queries.

Another way in which DNS servers co-operate with each other is by reducing the number of queries involved in resolving a domain name. When a DNS request is made for a domain name, the best matches for that domain name are sent in the response. For example, a DNS server holding cached information about the name server for `mit.edu` can direct the resolver requesting information about `www.mit.edu` to the authoritative server holding that information.

Caching is central to the efficiency of the DNS framework. Every internet-enabled computer is associated with a DNS server specific to its ISP (Internet Service Provider) or Intranet. This DNS server caches information accessed by its clients, thereby reducing the number of queries sent to authoritative servers, and reducing the time taken to handle each DNS query. Thus the DNS framework satisfies Requirement 2.

The DNS framework supports redundancy. There are multiple root DNS servers. Similarly, most domain names are managed by multiple DNS servers. Since each domain name can be associated with multiple NS addresses and A addresses, the DNS framework can be made redundant from the root servers to the Internet servers. Thus the DNS framework satisfies Requirements 3 and 4.

All DNS proposals and standards can be found in Request for Comments (RFCs) maintained by the Network Working Group. RFC 1034 and 1035 describe the basic concepts in DNS.

In this section, we have demonstrated that DNS satisfies Requirements 1 through 4 of the ONS framework. In the next section, we will show how ONS can be built on DNS and describe a method to satisfy Requirement 5 using translation format strings.

3.2. Translation Format Strings

In this section, we will describe how the ONS framework can be built on the DNS framework using **translation format strings**. In the previous section, we looked at the DNS framework and showed how ONS Requirements 1 through 4 are satisfied by the DNS framework. We will now see how an EPC can be formatted as a domain name during a **pre-resolution phase**. Once the EPC is converted to an **EPC domain name**, the DNS framework can be used to look up PML servers for the associated EPC. In the next section, we show that the translation format strings can be stored as TXT DNS records.

The EPC domain name consists of an **EPC domain prefix** that is computed from the EPC, and a constant suffix called the **EPC root domain**. The EPC root domain is a domain name chosen to be common domain suffix for ONS accesses chosen by the EPC namespace authority. Currently the EPC root domain is `epc.objid.net`.

Translation format strings are used to convert EPCs to EPC domain names during pre-resolution. A format string consists of digits 0 through 4 and the `'.'` character. Given an EPC and a format string, the following steps can be used to convert the EPC to an EPC domain prefix.

1. Every character in the format string is replaced with a character to obtain the EPC domain prefix from the given EPC.

2. A `'.'` or a `'0'` in the format string is translated to a `'.'` or a `'0'` (respectively) in the domain name. The character `'0'` is typically used to left-pad a label in the EPC domain prefix. The character `'.'` is used as a separator between labels.

3. The digit `'n'` in the format string, where n is between 1 and 4, is translated as a base (2^n) digit in the domain name as follows:

   a. The digit `'1'` is translated to a binary digit, which is either 0 or 1. This digit corresponds to one bit in the EPC.
   b. The digit `'2'` is translated to a quaternary digit, which is between 0 and 3. The digit corresponds to two bits in the EPC.
   c. The digit `'3'` is translated to an octal digit, which is between 0 and 7. The digit corresponds to three bits in the EPC.
   d. The digit `'4'` is translated to a hexadecimal digit, which is a character between `'0'` through `'9'` or `'A'` through `'F'`. The digit corresponds to four bits in the EPC.

4. The **bit-size** of the format string is the sum of digits present in the format string. For example, the bit-size of the format string `01.44.33.1.1.2` is (0+1+4+4+3+3+1+1+2), which is equal to 19.

5. If the EPC has fewer bits than the size of the format string, the transformation is said to be **erroneous.**

6. The EPC bits assigned to format digits `'1'` through `'4'` in the format string are as follows. The format string is parsed label-by-label from right to left. The digit `'n'` in the format string is assigned to the highest n unassigned bits in the EPC in the order in which they appear in the EPC.

The following observations can be made about the translation scheme described above.

1. The number of labels in the translated domain name is same as those in the format string.

2. Every label in the translated domain name is assigned to bits in the EPC in the order in which they appear in the EPC.

Translation format strings that are not erroneous can be categoriezed as complete and incomplete format strings.

Complete format string
A **complete format string** is one that specifies the translation scheme for all bits in the EPC.

Partial format string
A **partial format string** is one that specifies the translation scheme for only a prefix of the EPC.

For example, let us consider the partial format string `4.4.4444.1.1.1.3.3.3.013`, and the 64-bit EPC

`0110101001111001011010011100101111010001101011000110110101100100.`

Since the EPC has 64 bits, the least significant 24 bits are truncated to result in the EPC prefix

`0110101001111001011010011100101111010001.`

Figure 3 shows how the 40-bit EPC prefix is converted to an EPC domain prefix. Recall that the bits are assigned from the most significant to least significant bit in the EPC to the format string in the reverse order of labels. The first row shows the format string reversed label-by-label. The next row shows the 40-bit EPC prefix. The third row shows transformed digits in the reversed format string. On reversing the string in this row label-by-label, we obtain the EPC domain prefix.

Figure 3: Example: EPC domain translation based on a format string

EPC DOMAIN TRANSLATION BASED ON A FORMAT STRING

```
                        +---+---+-+---+-+---+-+---+-+-+-+-+-+-+-+-+----+----+----+----+-+----+-+-----+
Format String           |0|1|3   |.|3   |.|3   |.|3   |.|1|.|1|.|1|.|4    |4    |4    |4    |.|4    |.|4     |
reversed by label       | | |   | |   | |   | |   | | | | | | | | |    |    |    |    | |    | |     |
                        | | |   | |   | |   | |   | | | | | | | | |    |    |    |    | |    | |     |
Assigned bits in        | |0|110| |101| |001| |111| |0| |0| |1| |0110|1001|1100|1011| |1101| |0001 |
the EPC prefix          | | |   | |   | |   | |   | | | | | | | | |    |    |    |    | |    | |     |
                        | | |   | |   | |   | |   | | | | | | | | |    |    |    |    | |    | |     |
Reversed EPC            |0|0|6   |.|5   |.|1. | |7. | |0|.|0|.|1|.|6    |9    |C    |B    |.|D    |.|1     |
Domain                  | | |   | |   | |   | |   | | | | | | | | |    |    |    |    | |    | |     |
                        +-+-+---+-+---+-+---+-+---+-+-+-+-+-+-+-+-+----+----+----+----+-+----+-+-----+
```

Thus, the EPC domain prefix is `1.D.69CB.1.0.0.7.1.5.006` for the EPC prefix

`0110101001111001011010011100101111010001.`

The EPC domain can be obtained by appending the EPC root domain. For the above example, the EPC domain name is `1.D.69CB.1.0.0.7.1.5.006.epc.objid.net`

Translation format strings provide a general method for translating EPCs to EPC domain prefixes. Depending on the version of the EPC, we can use the appropriate translation format string to perform pre-resolution. Thus, Requirement 5 for the ONS framework can be satisfied.

One issue, however, is left unaddressed. How do we store the EPC format strings, allowing for the addition of new format strings for new EPC versions? The next section will present a solution to this issue.

3.3. Algorithm for Pre-resolution

In this section, we will show that translation format strings can be maintained in DNS TXT records. We will also look at the detailed algorithm for pre-resolution. Every ONS lookup will involve a pre-resolution followed by a standard DNS domain name resolution to obtain the IP addresses of PML servers.

Translation format strings can be stored in TXT records (refer to Section 3.1). This method has the advantage that the authoritative DNS server for that EPC domain suffix can maintain the translation format strings. Currently, we expect the EPC namespace authority to maintain translation format strings for each version of the EPC.

Although ONS resolvers are recommended to obtain translation format strings using DNS lookups, the authority governing the allocation of EPCs may choose to specify a complete translation format string for all EPCs belonging to a version. Such format strings are called **fixed format strings**. ONS resolvers can hard-code fixed translation format strings when performing an EPC to EPC domain name translation. For example, if the EPC namespace authority specifies a fixed translation format string for 64-bit EPCs with version 0, any ONS resolver implementation can directly compute the EPC domain prefix for EPCs having this version.

For versions that do not have fixed translation format strings, the translation format string lookup method shown below should be used.

Format translation strings can be obtained using DNS lookups on a special `info` record for the specific version of the EPC. Given an EPC, the format string `'44'` can be used to obtain the version number of the EPC, say `<version-number>`. A lookup on TXT records for `info.<version-number>.<root-domain>` will give the format string for the EPC under consideration. For example, the TXT record could look like

```
info.80.<root-domain> TXT 4444.4444.44.4444.2.2.2.2.444444.44
```

specifying a complete translation format string for a 96-bit EPC having version number 80 hex. The format string `'44'` that is used to get the EPC's version is hard-coded based on the bits appearing in the EPC.

The info record for an EPC version can contain

1. A complete format string
The ONS resolver can fully translate the EPC to an EPC domain. This method is called **direct translation**.

2. A partial format string
The ONS server will lookup

```
info.<incomplete-domain>.<root-domain>
```

record to get a refinement on the domain. This method is called **iterative translation**. Unlike direct translation, this method may require lookups in different ONS servers to translate the EPC.

Direct translation requires one DNS lookup to find the translation format string. The format strings are maintained in the `<root-domain>`. Since ONS resolvers cache records accessed during resolution, the TXT records specifying format strings will be cached for all versions that have been accessed, the cache will answer nearly all format string lookups.

Iterative translation requires more than one DNS lookup to find the translation format string. Again, the cache will answer most of these queries.

An example of iterative translation is given below. The EPC 01FAC38909 hex can be translated iteratively using the following DNS records:

1. Iter 1
`info.01.epc.objid.net TXT 4444.44`
(The partial EPC domain prefix is FAC3.01)

2. Iter 2

`info.FAC3.01.epc.objid.net TXT  4.2.2.4444.44`

(The partial EPC domain prefix is 9.0.2.FAC3.01)

3. Iter 3

`info.9.0.2.FAC3.01.epc.objid.net TXT  44.4.2.2.4444.44`

(The EPC domain prefix is 09.9.0.2.FAC3.01)

Thus, the EPC domain name is iteratively computed as 09.9.0.2.FAC3.01.epc.objid.net.

Iterative translation allows sub-domains to control the domain translation. This helps simplify the sub-domain's masking rules. For example, the three DNS records listed above could reside in three different name servers that have authority of the domains `epc.objid.net, 01.epc.objid.net` and `FAC3.01.epc.objid.net.`

On the other hand, direct translation simplifies ONS server management, and relies only on the EPC namespace authority to manage info records.

Iterative algorithm for ONS pre-resolver.
Given an EPC, the ONS resolver can locate its PML server as follows:

1. Based on the leading bits in the version number determine the format string to extract <version-number>
2. Compute the <version-number> based on the format string.
3. For global queries, use the <root-domain> `as epc.objid.net.` For local queries, use the pre-configured local root domain.
4. Lookup for TXT records in `info.<version-number>.<root-domain>`, where the <root-domain> is configured at the ONS resolver.
5. The last TXT record in the answer section of the DNS response (RFC 1034, 1035) will specify the format string to be used.
6. Translate the EPC to an EPC domain prefix based on the format string.
7. If the format string is partial, lookup for TXT records at `info.<partial-EPC-prefix>.<root-domain>`. and go to step 5.
8. If the format string is complete, the EPC domain name will be `info.<complete-EPC-prefix>.<root-domain>.`

Section 4.1 describes the implementation of the ONS resolver in detail.

4. IMPLEMENTATION DETAILS

This section describes the implementation details of the various components of the ONS architecture. The ONS architecture consists of the resolver and the server.

The ONS resolver uses the GNU adns library (http://www.chiark.greenend.org.uk/~ian/adns) to perform DNS lookups . Given an EPC and the EPC root domain, the ONS resolver computes the EPC domain name. Section 4.1 describes the ONS resolver.

The ONS server consists of a DNS server along with configuration tools to configure the DNS server to hold ONS mapping information.

ONS servers are implemented by appropriately configuring BIND (Berkeley Internet Name Domain) servers. BIND (Berkeley Internet Name Domain) is an open re-distributable reference implementation of the Domain Name System (DNS) protocols. BIND distributions can be obtained from the Internet Software Consortium (ISC) (http://www.isc.org).

In Section 4.2, we look at the basic configuration of secure BIND servers. Section 4.2 briefly describes the syntax and semantics of XML files specifying the server configuration.

This tool generates configuration files for BIND servers. The server configuration tool processes **ONS specification XML files** as input. Section 4.3 describes the ONS specification language in detail.

The configuration tool can be invoked using the command **onsconfig** to generate BIND configuration files. The ONS server can be started and stopped using the command **onsadmin**.

The configuration tool is written using XT (http://www.jclark.com/xml/xt.html) (an implementation of XSLT with some extensions). An XSLT file transforms the given XML specification files to BIND configuration files. XT extension functions allow the invocation of Java methods during the XSL transform. Furthermore, some simple **sed** scripts are used to format the input and output properly.

The ONS specification files can be updated using one of these methods:

1. Manual update
Administrative personnel can change the specification files. After changing any of these configuration files the ONS server must be restarted. This method is useful to manage top-level specification commands, such as the IP address of external and internal BIND server. This method is, however, not suited for managing large files with ONS entries.

2. Specification Management Tool
The XML file containing the ONS content file (ons-content) can be updated using the specification update tool, **onsupdate**. The syntax and semantics of the update file containing differences are described in Section 4.4.

3. Content Server.
The Content Server stores the mapping information in a database, and serves the specification XML files to one or more ONS servers. The Content Server is implemented as a webserver serving ONS Specification XML files over the HTTP protocol. Section 4.5 describes the Content Server.

To summarize,

1. ONS Resolver
The ONS resolver is implemented as an extension of the GNU adns library (http://www.chiark.greenend.org.uk /~ian/adns) C library.

2. DNS Server
BIND (http://www.isc.org) is used as the DNS Server, and is configured according to the secure BIND configuration guidelines described in Section 4.2

3. Server Configuration Tool
This tool processes ONS specification XML files as input. Section 4.3 describes the ONS specification language in detail.

4. Specification Management Tool
This tool updates ONS specification files given an ONS update file. The syntax and semantics of the update file containing differences are described in Section 4.4.

5. Content Server
This is a webserver serving ONS Specification XML files over the HTTP protocol based on the entries in the Content Server database. Section 4.5 describes the Content Server in detail.

4.1. The ONS Resolver

In this section, we describe the implementation of the ONS resolver. The ONS resolver is implemented using the GNU adns library (http://www.chiark.greenend.org.uk/~ian/adns) to perform DNS lookups.

Pre-resolution can be performed using methods implemented in the ONS resolver. The C data structure `ons_resolver_state` contains state information about the resolver initialization. This structure can be instantiated using the function `initialize_ons_resolver`, and used for pre-resolution in the function `get_epc_domain`.

```
/* Initialize the state data structure */
int initialize_ons_resolver(
    ons_resolver_state *state, /* Pointer to a declared state */
    const char *epc_domain_suffix, /* epc root domain */
    const char *resolver_conf, /* 0 if resolv.conf should be used,
                                  string holding resolv.conf ow. */
    int flags /* ORed value of flags in ons_flags */
);

/* Translate an epc to a domain name */
int get_epc_domain(
    ons_resolver_state state, /* Returned state */
    char *result_domain_name, /* 256 chars to hold the answer */
    int num_epc_bits, /* Number of bits in the epc */
    const char *epc, /* An array of chars containing the epc. */
                     /* The (n%8)th leading bit of epc[n/8] is */
                     /* the nth bit in the epc. */
);
```

A combined function `get_epc_domain_2` initializes the state and computes the EPC domain name of the given EPC. The function prototype is given below:

```
int get_epc_domain_2(
    char *result_domain_name,
    int num_epc_bits,
    const char *epc,
    const char *epc_domain_suffix,
    int flags);
```

The interface to ONS pre-resolver can be found in <onsresolver.h>. The functions can be found in the library ons. For example, a C program using ONS resolver utilities will include `<onsresolver.h>` and can be compiled using the command `gcc foo.c -lons`.

4.2. Secure BIND Configuration

In this section, we describe methods to ensure robust, secure and efficient configuration of BIND server.

The following steps have to be taken to ensure that the BIND server framework is robust:

1. Each resolver should be configured to contact with more than one DNS server. If one DNS server is down due to a crash, DNS restart or system upgrade, the resolver will contact other ONS servers.

2. Each zone in the ONS server should be associated with one or more **slaves**[2] that act as authoritative servers for that zone. These slaves perform **zone transfers** to update themselves with the latest authoritative information.

[2] Redundant BIND servers can be configured by assigning one or more slave servers to a master server

Some of the steps taken to ensure the BIND servers are secure are listed below:

1. BIND servers do not expose resolver functionality to the external world. This prevents "cache-poisoning" attacks.

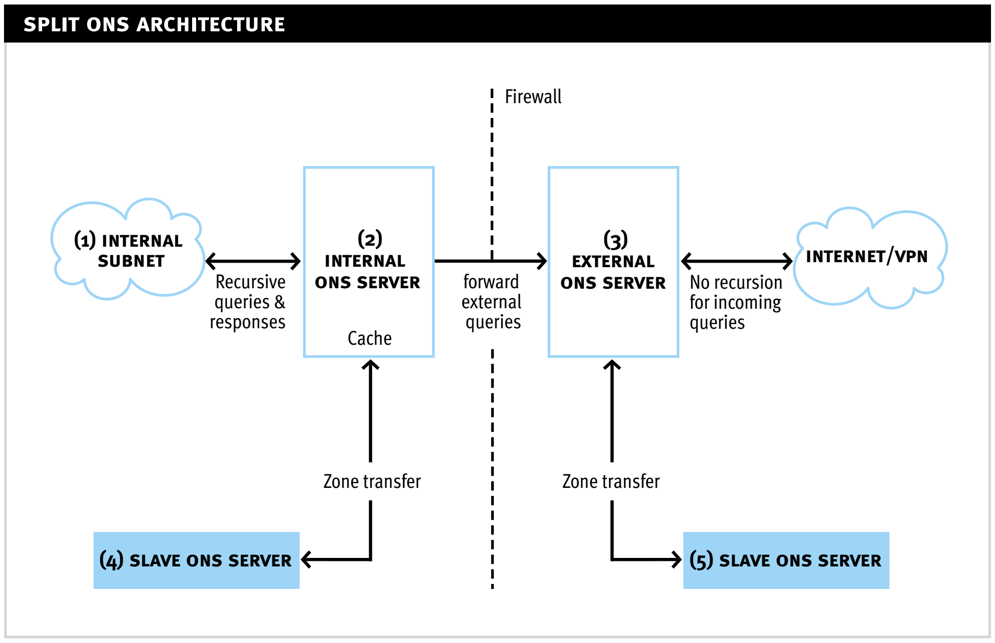
2. DNS records containing information for the internal network are not exposed to the external world.

3. Zone transfers should be limited to the slave machines. The TSIG protocol should be used to ensure that the transmitted DNS records are not modified in route.

The above requirements point to a **split-ONS architecture**. Each server consists of an internal server inside the firewall and an external server. The external server does not support resolver functionality for queries that do not come from the internal server. The internal server holds resource records meant for the internal network. Both these servers are held in user accounts with limited permissions, so that a hacker who gains control of the user account will not easily gain control of all the machine's resources.

Figure 4 describes the split-ONS server architecture.

Figure 4: Split-ONS architecture



To improve the efficiency of the ONS servers, all queries resolved by the internal ONS server are cached for future reference.

The next section describes the syntax and semantics of the XML configuration files.

4.3. The Specification Language

ONS specification files are written in XML syntax. In this section, we look at the parts of the DTD specification `ons-specification.dtd`.

The semantics of each tag are described in Appendix B along with its DTD syntax. A brief overview of each tag is given below.

ons-specification
The ons-specification tag is the top-level node for the ONS specification file. This tag contains the definition of the trusted subnet, the external ONS server, and an optional internal DNS server.

The attribute `name` gives a name to the specification. The attribute `translation-spec` holds a URL to the translation specification document containing information about various EPC versions used in the specification. The URL should refer to a document with the root tag `translation-spec`.

translation-spec
This is the root node for a translation specification file. The tag ons-specification contains the URL where this file is present.

A translation specification contains multiple version tags enclosed in it.

version
A version provides information about an EPC version. The attribute number should contain the version number. The attribute `version-format-string` gives the format string that returns the version number when applied on an EPC with that version. The attributes `length, version-length, manufacturer-length, product-length`, and `serial-length` contain the total, version field, manufacturer field, product field and serial field bit-length of EPCs in that version, respectively. The attribute format-string describes the `format string` to be used for EPCs with that version.

trusted-subnet
This tag contains IP masks (ip-mask) of addresses that can be allowed to access the internal BIND server, and use the BIND servers to perform recursive queries.

ip-mask
An IP mask is of the form:
```
<IP address mask>/<number of leading bits in the mask to compare
against the given IP address>.
```

For example, the IP mask 10.0.0.0/8 will match all IP address with the first 8 bits, i.e., the first byte matching 10, such as 10.1.120.255. Note that the IP address has 32 bits. The textual representation of an IP address consists of the decimal form of each byte separated by '.'s. Similarly, the IP mask 10.0.0.64/30 will match the IP addresses 10.0.0.64 through 10.0.0.67.

external
This tag contains the configuration information for the external DNS server when using split-ONS mode. The child tags admin, slaves, hint-file, zone-file, include-file, and ons-content-ref contain specific details about the external ONS server.

The attribute `ip` contains the IP address of external BIND server. The attribute `directory` corresponds to the directory in which all hint files and zone files can be found. The attribute `max-zone-transfer-time` specifies the time after which a zone transfer should be aborted. The attributes `pid-file, statistics-file` and `dump-file` correspond to the filenames of BIND's pid, statistics and dump files, respectively.

internal
This tag contains the configuration information for the internal DNS server when using split-ONS mode. The child tags admin, slaves, hint-file, zone-file, include-file, and ons-content-ref contain specific details about the external ONS server.

The attribute ip contains the IP address of internal BIND server. The attribute `directory` corresponds to the directory in which all hint files and zone files can be found. The attribute `max-zone-transfer-time` specifies the time after which a zone transfer should be aborted. The attributes `pid-file, statistics-file` and `dump-file` correspond to the filenames of BIND's pid, statistics and dump files, respectively.

admin
This tag contains information regarding the remote administration of the BIND server. The attribute port contains the `port` where the remote administration server will listen to. The utility rndc, provided in BIND distributions, can be used to administer the BIND server. The default port used is 953.

key
This tag contains information about secret keys that need to be used to perform remote administration. The attribute name has the `name` of the key. The same name should be given to key configured on the client side. The attribute `secret` contains the secret key. Secret keys can be generated using tools such as dnssec-gen. For example, a 512-bit secret key named `foo` can be generated for the hmac-md5 algorithm using dnssec-keygen utility provided in BIND distributions, as follows:

```
dnssec-keygen -a HMAC-MD5 -b 512 -n HOST rndc_key
```

The generated key should be copied into the secret attribute for the key.

slaves
This tag contains the allowed IP masks (ip-mask) that can perform zone transfers to the external or internal BIND server, depending on whether the tag is enclosed in the external or the internal tag.

hint-file
This tag contains information about hint files. The initial set of root nameservers is specified using a hint tag. The attribute `domain` is typically `'.'` corresponding to the root domain, and the attribute `file` contains the name of root hints file. The root hints file, usually named "db.cache", contains the IP addresses and domain names of available root servers.

zone-file
This tag contains information about DNS zone files that have to be included in the BIND server. The attribute `domain` holds the zone domain. The attribute `file` holds the filename of the zone file. This tag is used to hold DNS-related zone in the ONS server. For example, the ONS server could hold authoritative information about domains other than those in the ONS namespace.

include-file
This tag contains information about any file that has to be included in the BIND configuration file. The attribute file points to the `file` to be included.

ons-content-ref
The tag ons-content-ref holds the URL of a file having ons-content as the root tag, in the attribute `src`.

The location of the ONS content file can be an external URL. This feature is used by the ONS content server to physically separate the specification file from the content file for more centralized management of ONS servers.

ons-content
ONS content files hold authoritative information about the mapping between EPCs and IP addresses. This tag is the top-level tag for ONS content files. Master (master) and slave (slave) child tags specify the ONS zones managed by this server.

master
This tag contains authoritative information about a zone. A DNS zone consists of an SOA (Start of Authority) record followed by DNS resource records. A zone consists of an EPC range for which the server is authoritative and a set of EPC to IP address mapping held in that DNS zone.

This tag has exactly one authority definition (ons-authority) or a reference to an authority definition file (ons-authority-ref), followed by exactly one entry-set definition (ons-entries) or a reference to an entry-set definition file.

slave
This tag contains attributes that specify the slave zone. The attribute `master-ip` specifies the IP address of the master DNS server. The other attributes specify the appropriate EPC range. Refer to the tag epc-range regarding these attributes.

ons-authority
This tag is either enclosed in the master tag or placed as the root tag of a document referenced by an ons-authority-ref tag. This tag has information regarding the zone for which the BIND server is authoritative. Each master zone can be associated with multiple slave name servers using child slave-name-server tags.

Some attributes of this tag define the EPC range for which the zone is authoritative. These attributes can be found in the tag epc-range. The other attributes are:

1. TTL: Contains the default time-to-live for any resource record (RR) read from this server. If not specified, this attribute defaults to 1 day.

2. NEGATIVE-TTL: Contains the time for which any resource record (RR) that was not found in this BIND server can be assumed to be absent in this server. If not specified, this attribute defaults to 1 hour. This attribute determines how long it will take clients to recognize an ONS entry for a newly added PML server.

3. REFRESH-TIME: Contains the time after which a slave BIND server that performed a successful zone transfer should attempt another one. Defaults to 8 hours.

4. RETRY-TIME: Contains the time after which a slave BIND server that performed an unsuccessful zone transfer should attempt another one. Defaults to 2 hours.

5. EXPIRE-TIME: Contains the time after which a slave BIND server that performed the last successful zone transfer should not give authoritative information about the zone. Defaults to 4 weeks.

6. MASTER-NAME-SERVER: Contains the name of the master name server. This does not include the EPC domain corresponding to the EPC range. For example, the master name server will be `"NS1"`, if the master name server domain is `"NS1.00.pml.objid.net."`.

7. HOSTMASTER: Contains the email address of the host master for this zone. The `'@'` in the email address should be replaced with the `'.'` character. Consequently any `'.'`s in the email address preceding `'@'` should be replaced with the characters `'\.'`.

ons-authority-ref
This tag references an ONS authority file. The attribute `src` holds the URL to the ONS authority file.

ons-entries
This tag is either enclosed in the master tag or placed as the root tag of a document referenced by an ons-entries-ref tag.

This tag contains definitions of PML server hosts (defined by host tags) and ONS servers (defined by host tags). The attributes in this tag act as defaults for unspecified attributes in the enclosed EPC-range tags. Refer to the tag epc-range for the syntax and semantics of the attributes.

host
This tag contains the IP addresses of a set of PML servers (defined by ip tag) and the EPC ranges (defined by epc-range tag) for which the PML servers are applicable.

The attribute `id` is a unique identifier of that host or ns tag. The identifier is useful while performing updates using the Specification Management Tool. The attribute `ttl` can be used to specify a time-to-live value for the corresponding resource record that overrides the `ttl` value specified at the "ons-authority" tag.

ns
This tag contains the IP addresses of a set of ONS servers (defined by ip tag) and the EPC ranges (defined by epc-range tag)defining the sub-zones for which the servers are applicable.

The attribute `id` is a unique identifier of that host or ns tag. The identifier is useful while performing updates using the Specification Management Tool. The attribute `ttl` can be used to specify a time-to-live value for the corresponding resource record that overrides the `ttl` value specified at the "ons-authority" tag.

The attribute `name` specifies the name of the nameservers. As in the case of the tag slave-name-server, the name attribute specifies the domain name prefix of the name server.

ip
This tag holds the IP address of a PML server, in case the parent tag is a host tag, or the IP address of an ONS server, in case the parent tag is an ns tag.

epc-range
The epc-range tag is used to specify an EPC range for which a set of PML servers are applicable.

An EPC range is a range of EPCs that share the same leading bits. An EPC range is defined using the attributes `version, manufacturer, product, serial-number`.

The value of version should be the hex string corresponding to that version. The value of the manufacturer, product, serial number can be either an exact value in hex, or a hex value followed by a `'/'` and the number of leading bits that are masked. For example, the tag

```
<epc-range version="81" manufacturer="9AC8E0/20"/>
```

will map to all EPCs with version 81 and manufacturers with the first 20 bits as 9AC8E.

EPC ranges that specify the subset of all EPCs of the same version having the same leading bits are valid EPC ranges. For example,

```
<epc-range version="81" manufacturer="9AC8E0"
              product="9457A7" serial="A00000/10"/>
```

is a valid EPC range. However the EPC range

```
<epc-range version="81" manufacturer="9AC8E0"
              product="9457A0/20" serial="A00000"/>
```

is an invalid EPC range.

To summarize,

1. The ONS specification files are written in XML syntax. Appendix B describes the syntax and semantics of these tags.

2. The command **onsconfig** can be used to generate BIND configuration files given the ONS specification files.

3. Section 4.4 and Section 4.5 describe two methods to modify the ONS specification files.

4.4. The Content Update Language

This section describes the syntax and semantics of the XML input files to the Specification Management Tool. The Specification Management Tool (**onsupdate**) updates a specification content XML file, i.e., a file with root node ons-content using the Specification Management Tool. The DTD for the ONS Update Language can be found in `ons-specification.dtd.`

The semantics of each tag are described in Appendix C along with its DTD syntax. A brief overview of each tag is given below.

ons-content-update
This tag is the top-level node for an ONS content update file. This tag contains the definition of groups of update entries (defined by update-ons-entries) for zones in the ONS content file.

update-ons-entries
This tag contains the updates to be performed on an ONS zone. The attributes of this tag match the content file's ons-entries tag defining that zone. The child tags remove-host-or-ns, add, remove-from-host-or-ns and add-to-host-or-ns specify the update operations to be performed in ONS zone.

remove-host-or-ns
This tag removes a host or ns from the ONS zone defined by the parent update-ons-entries tag. The host or ns tag with the given `id` attribute is removed from the ONS zone.

add
This tag adds host or ns records to the ONS zone defined by the parent update-ons-entries tag. The enclosed host or ns tags, as defined in the ONS Specification Language are added to the ONS zone.

remove-from-host-or-ns
This tag removes EPC ranges or IP addresses from the host or ns tag in the ONS zone defined by the parent update-ons-entries tag. The child ip and epc-range tags, as defined in the ONS Specification Language are remove from the host or ns tag having the same `id` attribute, in that ONS zone.

add-to-host-or-ns
This tag adds EPC ranges and IP addresses to the host or ns tag in the ONS zone defined by the parent update-ons-entries tag. The child ip and epc-range tags, as defined in the ONS Specification Language are added to the host or ns tag having the same id attribute, in that ONS zone.

To summarize,

1. The ONS content update files are written in XML syntax. Appendix C describes the syntax and semantics of these tags.

2. The command **onsupdate** can be used to invoke the Specification Management Tool that updates ONS content files, i.e., a file with root node ons-content pointed by ons-content-ref in the ONS specification file.

4.5. The Content Server

In this section, we look at another tool to manage specification XML files using a database as the backend. This system works as follows:

1. The tag ons-content-ref points to an http or https address at the localhost or a remote machine.
2. The content server running on this machine will serve the ons-content files to the configuration tool when it is executed.

The content server consists of the following components:

1. A database holding records related to the zones held in the ONS server.
2. A Java class that read the database and compose the content XML files.
3. A cgi-bin wrapper that calls this Java class to serve the XML files.

We will now look at the database schema for the Content Server. The database schema consists of the following tables, each corresponding to a tag in the XML document. Each table has a numeric primary key that can be loaded from a sequence named `<table-name>_seq,` and columns representing attributes of the corresponding tag. Each table also has a numeric foreign key to the table that corresponds to the parent tag. The root table is called ons_client.

The table `ons_client` contains the various client machines running the ONS server configuration tool. Each client is given a name, which as to be passed as a GET variable while looking up the content server. Specifically, the ons-content-ref at the client machine will have its `src` attribute as

```
http://<content-server>/<cgi-script>?ONSCLIENTNAME=<client-name>
```

The other tables in the schema are described in Appendix D. Figure D-1 shows diagrammatic representation of the schema.

4.6. Administration Utilities

In this section, we describe all the administration utilities provided in the ONS package.

The ONS server package contains the administration utilities, the server configuration tool and the specification management tool. INSTALL and README files in the root directory of the package contain instructions regarding the configuration of the ONS server package. Every administration utility prints instructions on its use when invoked with no arguments.

The administration utilities provided with the ONS package are as follows:

1. **onsadmin:** Startup or shutdown the ONS server.
2. **onslookup:** Invoke the pre-resolver given the EPC bit string and domain suffix.
3. **onsconfig:** Generate the BIND configuration files using the server configuration tool.
4. **onsupdate:** Update a specification content XML file using the specification management tool.

4.7. Performance Statistics

The overhead of ONS lookups depends heavily on the network organization. The network latency is typically the bottleneck in an ONS lookup, unless the local ONS server is accessible over a LAN (Local Area Network). If a local ONS server is configured within the LAN, it will cache the `info` records accessed by ONS resolver.

On a DELL PowerEdge Server 2500, with 1133MHz Intel Pentium III processor, 512KB cache, and 512MB RAM, running BIND 9.1.3, an average ONS lookup took **0.27 ms**. This measure was approximately the same when this server was accessed by other machines its 100Mbps ethernet LAN.

The same server, when configured to use the DNS server provided by the ISP, took tens of milliseconds to respond to ONS lookups.

5. CONCLUSION

The Object Name Service package implements the following components:

1. An ONS pre-resolver implemented as a C library packaged with GNU adns. The pre-resolver converts EPC addresses to EPC domain names, based on formatting information stored as DNS resource records. These domains can be searched using a normal DNS lookup to obtain the IP address of the associated PML server.

2. A Server Configuration Tool that translates ONS specification files syntactically described by the XML DTD `ons-specification.dtd` to configuration files for BIND 9 servers. The BIND servers are configured to be secure, robust and efficient. The server configuration tool is implemented in XT and Java.

3. A Specification Management Tool that processes XML specification update files and appropriately updates specification files. The syntax for XML update files can also be found in `ons-specification.dtd.`

4. A Content Server that emits XML specification files based on the entries in a database. Unlike, the specification management tool, the specification file is regenerated by the Content Server on every execution of the Server Configuration Tool.

The ONS package also contains a set of administration utilities to perform the following tasks:

1. **onsadmin:** Startup or shutdown the ONS server.
2. **onslookup**: Invoke the pre-resolver given the EPC bit string and domain suffix.
3. **onsconfig:** Generate the BIND configuration files using the server configuration tool.
4. **onsupdate:** Update a specification content XML file using the specification management tool.

A. ELECTRONIC PRODUCT CODES (EPCS)

The EPC namespace is hierarchically assigned for ease in management. An Electronic Product Code (EPC) is of the form

<version number> . <manufacturer ID> . <product ID> . <serial number>,

where
- **version number**, also called the **primary version** number is a constant that specifies the format of EPC. The version number is 2 bits long if the EPC does not begin with the bits 00. Otherwise the version is 8 bits long.
- **manufacturer ID** is a bit string identifying the manufacturer
- **product ID** is a bit string identifying the product provided by the manufacturer
- **serial number** is a bit string identifying the serial number of the product provided by the manufacturer

Each version defines the bit-length of the manufacturer ID, product ID and serial number.

The **EPC namespace authority** will assign manufacturers with unique manufacturer IDs. The manufacturers can then independently assign unique product IDs to their products. The manufacturers or product-specific divisions of the manufacturers can then assign unique serial numbers to various instances of a product.

The number of bits in the EPC tag affects the cost of the tag. A trade-off should therefore be established between the length of the EPC and the cost of the tag. Since this trade-off may not be the same in future, tags with longer bit length can become viable. Consequently, the EPC reserves 8-bits for identifying the version.

B. ONS SPECIFICATION LANGUAGE REFERENCE

This section describes each tag in the ONS Specification Language.

ons-specification

Name
`ons-specification` — Root node for the specification file

Synopsis
PARENT TAGS. None
CHILD TAGS. trusted-subnet, external, and internal.
DTD.

```
<!ELEMENT ons-specification (trusted-subnet, external, internal?)>

<!ATTLIST ons-specification
       name CDATA #REQUIRED
       translation-spec CDATA #REQUIRED>
```

Description
The ons-specification tag is the top-level node for the ONS specification file. This tag contains the definition of the trusted subnet, the external ONS server, and an optional internal DNS server.

The attribute `name` gives a name to the specification. The attribute `translation-spec` holds a URL to the translation specification document containing information about various EPC versions used in the specification. The URL should refer to a document with the root tag `translation-spec`.

translation-spec

Name
`translation-spec` — Root node for the translation specification file

Synopsis
PARENT TAGS. None
CHILD TAGS. translation-spec
DTD.

```
<!ELEMENT translation-spec (version)*>
```

Description
This is the root node for a translation specification file. The tag ons-specification contains the URL where this file is present.

A translation specification contains multiple version tags enclosed in it.

version

Name

`version` — Specifies the translation format string information for a version

Synopsis

PARENT TAGS. translation-spec
CHILD TAGS. None
DTD.

```
<!ELEMENT version EMPTY>
<!ATTLIST version
       number CDATA #REQUIRED
       version-format-string CDATA #REQUIRED
       length CDATA #REQUIRED
       version-length CDATA #REQUIRED
       manufacturer-length CDATA #REQUIRED
       product-length CDATA #REQUIRED
       serial-length CDATA #REQUIRED
       format-string CDATA #REQUIRED
>
```

Description

A version provides information about an EPC version. The attribute number should contain the version number. The attribute `version-format-string` gives the format string that returns the version number when applied on an EPC with that version. The attributes `length, version-length, manufacturer-length, product-length`, and `serial-length` contain the total, version field, manufacturer field, product field and serial field bit-length of EPCs in that version, respectively. The attribute format-string describes the `format string` to be used for EPCs with that version.

trusted-subnet

Name

`trusted-subnet` — Defines a trusted subnet for a split-ONS server

Synopsis

PARENT TAGS. ons-specification
CHILD TAGS. ip-mask
DTD.

```
<!ELEMENT trusted-subnet (ip-mask)*>
```

Description

This tag contains IP masks (ip-mask) of addresses that can be allowed to access the internal BIND server, and use the BIND servers to perform recursive queries.

ip-mask

Name
`ip-mask` — Defines an IP address mask

Synopsis
PARENT TAGS. trusted-subnet, admin, and slaves
CHILD TAGS. None
DTD.

```
<!ELEMENT ip-mask (#PCDATA) >
```

Description
An IP mask is of the form:
`<IP address mask>/<number of leading bits in the mask to compare against the given IP address>.`

For example, the IP mask 10.0.0.0/8 will match all IP address with the first 8 bits, i.e., the first byte matching 10, such as 10.1.120.255. Note that the IP address has 32 bits. The textual representation of an IP address consists of the decimal form of each byte separated by `'.'`s. Similarly, the IP mask 10.0.0.64/30 will match the IP addresses 10.0.0.64 through 10.0.0.67.

external

Name
`external` — Contains configuration information for the external ONS server

Synopsis
PARENT TAGS. ons-specification
CHILD TAGS. admin, slaves, hint-file, zone-file, include-file, and ons-content-ref
DTD.

```
<!ELEMENT external (admin?, slaves?, hint-file*, zone-file*, include-file*,
ons-content-ref*)>
<!ATTLIST external
       ip CDATA #REQUIRED
       directory CDATA "/var/named"
       pid-file CDATA "/var/run/named.pid"
       statistics-file CDATA "/var/run/named.stats"
       dump-file CDATA "/var/run/named.dump"
       max-zone-transfer-time CDATA "60"
>
```

Description
This tag contains the configuration information for the external DNS server when using split-ONS mode. The child tags admin, slaves, hint-file, zone-file, include-file, and ons-content-ref contain specific details about the external ONS server.

The attribute `ip` contains the IP address of external BIND server. The attribute `directory` corresponds to the directory in which all hint files and zone files can be found. The attribute `max-zone-transfer-time` specifies the time after which a zone transfer should be aborted. The attributes `pid-file, statistics-file` and `dump-file` correspond to the filenames of BIND's pid, statistics and dump files, respectively.

internal

Name

`internal` — Contains configuration information for the internal ONS server

Synopsis

PARENT TAGS. ons-specification
CHILD TAGS. admin, slaves, hint-file, zone-file, include-file, and ons-content-ref
DTD.

```
<!ELEMENT internal (admin?, slaves?, hint-file*, zone-file*, include-file*,
ons-content-ref*)>
<!ATTLIST internal
      ip CDATA #REQUIRED
      directory CDATA "/var/named"
      pid-file CDATA "/var/run/named.pid"
      statistics-file CDATA "/var/run/named.stats"
      dump-file CDATA "/var/run/named.dump"
      max-zone-transfer-time CDATA "60"
>
```

Description

This tag contains the configuration information for the internal DNS server when using split-ONS mode. The child tags admin, slaves, hint-file, zone-file, include-file, and ons-content-ref contain specific details about the external ONS server.

The attribute `ip` contains the IP address of internal BIND server. The attribute `directory` corresponds to the directory in which all hint files and zone files can be found. The attribute `max-zone-transfer-time` specifies the time after which a zone transfer should be aborted. The attributes `pid-file, statistics-file` and `dump-file` correspond to the filenames of BIND's pid, statistics and dump files, respectively.

admin

Name
`admin` — Specify remote administration details

Synopsis
PARENT TAGS. external, and internal
CHILD TAGS. key, and ip-mask
DTD.

```
<!ELEMENT admin (key+, ip-mask+)>
<!ATTLIST admin
       port CDATA "953"
>
```

Description
This tag contains information regarding the remote administration of the BIND server. The attribute `port` contains the port where the remote administration server will listen to. The utility rndc, provided in BIND distributions, can be used to administer the BIND server. The default port used is 953.

key

Name
`key` — Defines a secret key

Synopsis
PARENT TAGS. admin
CHILD TAGS. None
DTD.

```
<!ELEMENT key EMPTY>
<!ATTLIST key
       name CDATA #REQUIRED
       algorithm CDATA #REQUIRED
       secret CDATA #REQUIRED
>
```

Description
This tag contains information about secret keys that need to be used to perform remote administration. The attribute `name` has the name of the key. The same name should be given to key configured on the client side. The attribute secret contains the `secret` key. Secret keys can be generated using tools such as dnssec-gen. For example, a 512-bit secret key named `foo` can be generated for the hmac-md5 algorithm using dnssec-keygen utility provided in BIND distributions, as follows:

```
dnssec-keygen -a HMAC-MD5 -b 512 -n HOST rndc_key
```

The generated key should be copied into the secret attribute for the key.

slaves

Name
`slaves` — Defines the allowed slave ONS servers

Synopsis
PARENT TAGS. external, and internal
CHILD TAGS. ip-mask
DTD.

```
<!ELEMENT slaves (ip-mask)* >
```

Description
This tag contains the allowed IP masks (ip-mask) that can perform zone transfers to the external or internal BIND server, depending on whether the tag is enclosed in the external or the internal tag.

hint-file

Name
`hint-file` — Points to the hint file containing the set of root server domain names and IP addresses

Synopsis
PARENT TAGS. external, and internal
CHILD TAGS. None
DTD.

```
<!ELEMENT hint-file EMPTY >
<!ATTLIST hint-file
       domain CDATA #REQUIRED
       file CDATA #REQUIRED
>
```

Description
This tag contains information about hint files. The initial set of root nameservers is specified using a hint tag. The attribute `domain` is typically `'.'` corresponding to the root domain, and the attribute `file` contains the name of root hints file. The root hints file, usually named "db.cache", contains the IP addresses and domain names of available root servers.

zone-file

Name
`zone-file` — Points to DNS zone files

Synopsis
PARENT TAGS. external, and internal
CHILD TAGS. None
DTD.

```
<!ELEMENT zone-file EMPTY >
<!ATTLIST zone-file
       domain CDATA #REQUIRED
       file CDATA #REQUIRED
>
```

Description
This tag contains information about DNS zone files that have to be included in the BIND server. The attribute `domain` holds the zone domain. The attribute `file` holds the filename of the zone file.

This tag is used to hold DNS-related zone in the ONS server. For example, the ONS server could hold authoritative information about domains other than those in the ONS namespace.

include-file

Name
`include-file` — Points to a BIND include file

Synopsis
PARENT TAGS. external, and internal
CHILD TAGS. None
DTD.

```
<!ELEMENT include-file EMPTY >
<!ATTLIST include-file
       file CDATA #REQUIRED
>
```

Description
This tag contains information about any file that has to be included in the BIND configuration file. The attribute `file` points to the file to be included.

ons-content-ref

Name
`ons-content-ref` — Points to the content file specifying mapping between EPCs and PML server addresses

Synopsis
PARENT TAGS. external, and internal
CHILD TAGS. None
DTD.

```
<!ELEMENT ons-content-ref EMPTY >
<!ATTLIST ons-content-ref
      src CDATA #REQUIRED
>
```

Description
The tag ons-content-ref holds the URL of a file having ons-content as the root tag, in the attribute `src`.

The location of the ONS content file can be an external URL. This feature is used by the ONS content server to physically separate the specification file from the content file for more centralized management of ONS servers.

ons-content

Name
`ons-content` — Root tag for ONS content files

Synopsis
PARENT TAGS. None
CHILD TAGS. master, and slave
DTD.

```
<!ELEMENT ons-content (master, slave)* >
```

Description
ONS content files hold authoritative information about the mapping between EPCs and IP addresses. This tag is the top-level tag for ONS content files. Master (master) and slave (slave) child tags specify the ONS zones managed by this server.

master

Name
`master` — Contains authoritative information about an ONS zone

Synopsis
PARENT TAGS. ons-content
CHILD TAGS. ons-authority, ons-authority-ref, ons-entries, and ons-entries-ref
DTD.

```
<!ELEMENT master ((ons-authority|ons-authority-ref), (ons-entries|ons-
entries-ref)) >
```

Description
This tag contains authoritative information about a zone. A DNS zone consists of an SOA (Start of Authority) record followed by DNS resource records. A zone consists of an EPC range for which the server is authoritative and a set of EPC to IP address mapping held in that DNS zone.

This tag has exactly one authority definition (ons-authority) or a reference to an authority definition file (ons-authority-ref), followed by exactly one entry-set definition (ons-entries) or a reference to an entry-set definition file.

slave

Name
`slave` — Contains master-server information about a slave zone

Synopsis
PARENT TAGS. ons-content
CHILD TAGS. None
DTD.

```
<!ELEMENT slave EMPTY >
<!ATTLIST slave
      version CDATA #REQUIRED
      manufacturer CDATA ""
      product CDATA ""
      serial CDATA ""
      domain-suffix CDATA #REQUIRED
      master-ip CDATA #REQUIRED
>
```

Description
This tag contains attributes that specify the slave zone. The attribute `master-ip` specifies the IP address of the master DNS server. The other attributes specify the appropriate EPC range. Refer to the tag epc-range regarding these attributes.

ons-authority

Name

`ons-authority` — Contains information regarding a zone for which the ONS server is authoritative

Synopsis

PARENT TAGS. master
CHILD TAGS. slave-name-server
DTD.

```
<!ELEMENT ons-authority (slave-name-server*) >
<!ATTLIST ons-authority
       ttl CDATA "1D"
       negative-ttl CDATA "1H"
       refresh-time CDATA "8H"
       retry-time CDATA "2H"
       expire-time CDATA "4W"
       master-name-server CDATA #REQUIRED
       hostmaster CDATA #REQUIRED
       version CDATA #REQUIRED
       manufacturer CDATA ""
       product CDATA ""
       serial CDATA ""
       domain-suffix CDATA #REQUIRED
>
```

Description

This tag is either enclosed in the master tag or placed as the root tag of a document referenced by an ons-authority-ref tag. This tag has information regarding the zone for which the BIND server is authoritative. Each master zone can be associated with multiple slave name servers using child slave-name-server tags.

Some attributes of this tag define the EPC range for which the zone is authoritative. These attributes can be found in the tag epc-range. The other attributes are:

1. TTL: Contains the default time-to-live for any resource record (RR) read from this server. If not specified, this attribute defaults to 1 day.

2. NEGATIVE-TTL: Contains the time for which any resource record (RR) that was not found in this BIND server can be assumed to be absent in this server. If not specified, this attribute defaults to 1 hour. This attribute determines how long it will take clients to recognize an ONS entry for a newly added PML server.

3. REFRESH-TIME: Contains the time after which a slave BIND server that performed a successful zone transfer should attempt another one. Defaults to 8 hours.

4. RETRY-TIME: Contains the time after which a slave BIND server that performed an unsuccessful zone transfer should attempt another one. Defaults to 2 hours.

5. EXPIRE-TIME: Contains the time after which a slave BIND server that performed the last successful zone transfer should not give authoritative information about the zone. Defaults to 4 weeks.

6. MASTER-NAME-SERVER: Contains the name of the master name server. This does not include the EPC domain corresponding to the EPC range. For example, the master name server will be `"NS1"`, if the master name server domain is `"NS1.00.pml.objid.net."`.

7. HOSTMASTER: Contains the email address of the host master for this zone. The `'@'` in the email address should be replaced with the `'.'` character. Consequently any '.'s in the email address preceding `'@'` should be replaced with the characters `'\.'`.

ons-authority-ref

Name

`ons-authority-ref` — URL reference to an ONS authority file

Synopsis

PARENT TAGS. master
CHILD TAGS. None
DTD.

```
<!ELEMENT ons-authority-ref EMPTY >
<!ATTLIST ons-authority-ref
       src CDATA #REQUIRED
>
```

Description

This tag references an ONS authority file. The attribute `src` holds the URL to the ONS authority file.

ons-entries

Name

`ons-entries` — Defines the mappings for PML servers and ONS servers

Synopsis

PARENT TAGS. master
CHILD TAGS. host, and ns
DTD.

```
<!ELEMENT ons-entries (host|ns)* >
<!ATTLIST ons-entries
       version CDATA #REQUIRED
       manufacturer CDATA ""
       product CDATA ""
       serial CDATA ""
       domain-suffix CDATA #REQUIRED
>
```

Description
This tag is either enclosed in the master tag or placed as the root tag of a document referenced by an ons-entries-ref tag.

This tag contains definitions of PML server hosts (defined by host tags) and ONS servers (defined by host tags). The attributes in this tag act as defaults for unspecified attributes in the enclosed epc-range tags. Refer to the tag epc-range for the syntax and semantics of the attributes.

host

Name
`host` — Contains information about a set of PML servers

Synopsis
PARENT TAGS. ons-entries
CHILD TAGS. ip, and epc-range
DTD.

```
<!ELEMENT host (ip*, epc-range*) >
<!ATTLIST host
       id  CDATA #REQUIRED
       ttl CDATA "1D"
>
```

Description
This tag contains the IP addresses of a set of PML servers (defined by ip tag) and the EPC ranges (defined by epc-range tag) for which the PML servers are applicable.

The attribute `id` is a unique identifier of that host or ns tag. The identifier is useful while performing updates using the Specification Management Tool. The attribute `ttl` can be used to specify a time-to-live value for the corresponding resource record that overrides the `ttl` value specified at the "ons-authority" tag.

ns

Name
`ns` — Contains information about a set of ONS servers responsible for a sub-zone

Synopsis
PARENT TAGS. ons-entries
CHILD TAGS. ip, and epc-range
DTD.

```
<!ELEMENT ns (ip*, epc-range*) >
<!ATTLIST host
       id  CDATA #REQUIRED
       name CDATA #REQUIRED
       ttl CDATA "1D"
>
```

Description
This tag contains the IP addresses of a set of ONS servers (defined by ip tag) and the EPC ranges (defined by epc-range tag)defining the sub-zones for which the servers are applicable.

The attribute `id` is a unique identifier of that host or ns tag. The identifier is useful while performing updates using the Specification Management Tool. The attribute `ttl` can be used to specify a time-to-live value for the corresponding resource record that overrides the `ttl` value specified at the "ons-authority" tag.

The attribute `name` specifies the name of the nameservers. As in the case of the tag slave-name-server, the name attribute specifies the domain name prefix of the name server.

ip

Name
`ip` — Contains the IP address of a PML server or ONS server

Synopsis
PARENT TAGS. host, ns, remove-from-host-or-ns, and add-to-host-or-ns
CHILD TAGS. None
DTD.

```
<!ELEMENT ip (#PCDATA) >
```

Description
This tag holds the IP address of a PML server, in case the parent tag is a host tag, or the IP address of an ONS server, in case the parent tag is an ns tag.

epc-range

Name

`epc-range` — Specifies an EPC range

Synopsis

PARENT TAGS. host, ns, remove-from-host-or-ns, and add-to-host-or-ns
CHILD TAGS. None
DTD.

```
<!ELEMENT epc-range EMPTY>
<!ATTLIST epc-range
       version CDATA ""
       manufacturer CDATA ""
       product CDATA ""
       serial CDATA ""
>
```

Description

The epc-range tag is used to specify an EPC range for which a set of PML servers are applicable.

An EPC range is a range of EPCs that share the same leading bits. An EPC range is defined using the attributes `version, manufacturer, product, serial-number.`

The value of version should be the hex string corresponding to that version. The value of the manufacturer, product, serial number can be either an exact value in hex, or a hex value followed by a `'/'` and the number of leading bits that are masked. For example, the tag

```
<epc-range version="81" manufacturer="9AC8E0/20"/>
```

will map to all EPCs with version 81 and manufacturers with the first 20 bits as 9AC8E.

EPC ranges that specify the subset of all EPCs of the same version having the same leading bits are valid EPC ranges. For example,

```
<epc-range version="81" manufacturer="9AC8E0"
               product="9457A7" serial="A00000/10"/>
```

is a valid EPC range. However the EPC range

```
<epc-range version="81" manufacturer="9AC8E0"
               product="9457A0/20" serial="A00000"/>
```

is an invalid EPC range.

C. ONS CONTENT UPDATE LANGUAGE REFERENCE

This section describes each tag in the ONS Content Update Language.

ons-content-update

Name
`ons-content-update` — Root node for the content update file

Synopsis
PARENT TAGS. None
CHILD TAGS. update-ons-entries
DTD.

```
<!ELEMENT ons-content-update (update-ons-entries)* >
```

Description
This tag is the top-level node for an ONS content update file. This tag contains the definition of groups of update entries (defined by update-ons-entries) for zones in the ONS content file.

update-ons-entries

Name
`update-ons-entries` — Updates an ONS zone in the input content file

Synopsis
PARENT TAGS. ons-content-update
CHILD TAGS. remove-host-or-ns, add, remove-from-host-or-ns and add-to-host-or-ns
DTD.

```
<!ELEMENT update-ons-entries (remove-host-or-ns, add, remove-from-host-or-
ns, add-to-host-or-ns) >
<!ATTLIST update-ons-entries
       version CDATA #REQUIRED
       manufacturer CDATA ""
       product CDATA ""
       serial CDATA ""
       domain-suffix CDATA #REQUIRED
>
```

Description
This tag contains the updates to be performed on an ONS zone. The attributes of this tag match the content file's ons-entries tag defining that zone. The child tags remove-host-or-ns, add, remove-from-host-or-ns and add-to-host-or-ns specify the update operations to be performed in ONS zone.

remove-host-or-ns

Name

`remove-host-or-ns` — Removes a host or ns from an ONS zone.

Synopsis

PARENT TAGS. update-ons-entries
CHILD TAGS. None
DTD.

```
<!ELEMENT remove-host-or-ns EMPTY >
<!ATTLIST remove-host-or-ns
       id CDATA #REQUIRED
>
```

Description

This tag removes a host or ns from the ONS zone defined by the parent update-ons-entries tag.
The host or ns tag with the given `id` attribute is removed from the ONS zone.

add

Name

`add` — Adds host or ns records to an ONS zone

Synopsis

PARENT TAGS. update-ons-entries
CHILD TAGS. None
DTD.

```
<!ELEMENT add (host|ns)* >
```

Description

This tag adds host or ns records to the ONS zone defined by the parent update-ons-entries tag.
The enclosed host or ns tags, as defined in the ONS Specification Language are added to the ONS zone.

remove-from-host-or-ns

Name

`remove-from-host-or-ns` — Removes EPC ranges an IP addresses from the host or ns tag in an ONS zone

Synopsis

PARENT TAGS. update-ons-entries
CHILD TAGS. None
DTD.

```
<!ELEMENT remove-from-host-or-ns (epc-range | ip)* >
<!ATTLIST remove-from-host-or-ns
       id CDATA #REQUIRED
>
```

Description

This tag removes EPC ranges or IP addresses from the host or ns tag in the ONS zone defined by the parent update-ons-entries tag. The child ip and epc-range tags, as defined in the ONS Specification Language are remove from the host or ns tag having the same `id` attribute, in that ONS zone.

add-to-host-or-ns

Name

`add-to-host-or-ns` — Adds EPC ranges and IP addresses to the host or ns tag in an ONS zone

Synopsis

PARENT TAGS. update-ons-entries
CHILD TAGS. None
DTD.

```
<!ELEMENT add-to-host-or-ns (epc-range | ip)* >
<!ATTLIST add-to-host-or-ns
       id CDATA #REQUIRED
>
```

Description

This tag adds EPC ranges and IP addresses to the host or ns tag in the ONS zone defined by the parent update-ons-entries tag. The child ip and epc-range tags, as defined in the ONS Specification Language are added to the host or ns tag having the same `id` attribute, in that ONS zone.

D. CONTENT SERVER DATABASE SCHEMA

This section describes the ONS Content Server schema. Figure D-1 depicts tables and columns in the schema. Each table in the schema is discussed below.

Figure D-1: Schema for the content server

SCHEMA FOR THE CONTENT SERVER

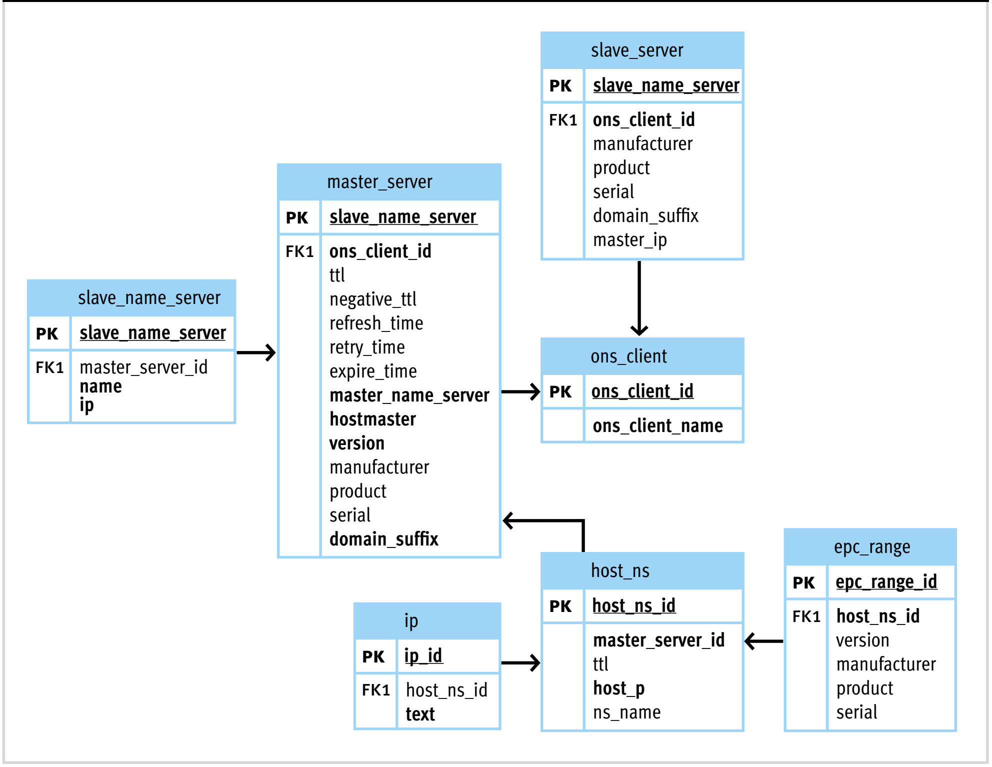


Table: `slave_server`
Tag: slave
Attribute columns: manufacturer, product, serial, domain-suffix, master-ip
Comments: Each slave zone specified in this table is linked to the table ons_client using the column ons_client_id.

Table: `master_server`
Tag: master, ons-authority, ons-entries
Attribute columns: ttl, negative_ttl, refresh_time, retry_time, expire_time, master_name_server, hostmaster, version, manufacturer, product, serial, domain_suffix
Comments: Each master zone specified in this table is linked to the table ons_client using the column ons_client_id. This tag also contains attributes of the ons-authority and ons-entries tags that are children of the master tag.

Table: `slave_name_server`
Tag: slave-name-server
Attribute columns: name, ip
Comments: Each slave name server is associated with its master zone using the foreign key master_server_id to the table master_server.

Table: `host_ns`
Tag: host OR ns
Attribute columns: ttl, ns_name
Comments: Each host_ns record is associated with its master_server record using the foreign key master_server_id. The column host_p is set to either "T" or "F" depending on whether the record is for a host or a nameserver. The column ns_name is null for hosts and contains the tag name for nameservers.
Table: `epc_range`
Tag: epc-range
Attribute columns: version, manufacturer, product, serial
Comments: Each epc_range record is associated with its host_ns record using the foreign key host_ns_id.

Table: `ip`
Tag: ip
Attribute columns: text
Comments: The column text corresponds to the text in the IP tag, which corresponds to one IP address of the host or ns to which this record points using the foreign key host_ns_id.

Designed and produced by Foxner. www.foxner.com