NENA Technical Information Document

On

XML Namespaces

NENA Technical Information Document on XML Namespaces
NENA 02-503, Issue 1, February 23, 2007

Prepared by: Pierre Desjardins, Positron Inc.
For National Emergency Number Association (NENA) DTC XML Working Group

Published by NENA
Printed in USA

NENA
TECHNICAL INFORMATION DOCUMENT

NOTICE

The National Emergency Number Association (NENA) publishes this document as an information source for the designers and manufacturers of systems to be utilized for the purpose of processing emergency calls. It is not intended to provide complete design specifications or parameters or to assure the quality of performance for systems that process emergency calls.

NENA reserves the right to revise this TID for any reason including, but not limited to:

- conformity with criteria or standards promulgated by various agencies
- utilization of advances in the state of the technical arts
- or to reflect changes in the design of network interface or services described herein.

It is possible that certain advances in technology will precede these revisions. Therefore, this TID should not be the only source of information used. NENA recommends that members contact their Telecommunications Carrier representative to ensure compatibility with the 9-1-1 network.

Patents may cover the specifications, techniques, or network interface/system characteristics disclosed herein. No license expressed or implied is hereby granted. This document shall not be construed as a suggestion to any manufacturer to modify or change any of its products, nor does this document represent any commitment by NENA or any affiliate thereof to purchase any product whether or not it provides the described characteristics.

This document has been prepared solely for the voluntary use of E9-1-1 Service System Providers, network interface and system vendors, participating telephone companies, etc.

By using this document, the user agrees that NENA will have no liability for any consequential, incidental, special, or punitive damages arising from use of the document.

NENA's Technical Committee has developed this document. Recommendations for change to this document may be submitted to:

National Emergency Number Association
4350 North Fairfax Drive, Suite750
Arlington, VA 22203-1695
800-332-3911
or: techdoccomments@nena.org

Acknowledgments:

The National Emergency Number Association (NENA) Data Technical Committee developed this document.

NENA recognizes the following industry experts and their companies for their contributions in development of this document.

Members:	**Company**
Delaine Arnold – DTC Chair	Independent Consultant
Kim Leigh – XML WG Leader	Qwest
Pierre Desjardins – Author/Editor	Positron

TABLE OF CONTENTS

1 EXECUTIVE OVERVIEW 5

1.1 Purpose and Scope of Document 5
1.2 Reason for Issue 5
1.3 Reason for Reissue 5
1.4 Recommendation for Standards Development work 5
1.5 Costs Factors 6
1.6 Acronyms/Abbreviations 6
1.7 Intellectual Property Rights Policy 6
1.7.1 General Policy Statement 6

2 INTRODUCTION TO THE CONCEPT OF *NAMESPACE* 7

2.1 General concept 7
2.2 Moving on to XML 7

3 XML NAMESPACES 9

3.1 Motivation for using namespaces in XML 9
3.2 Namespace declarations 9
3.2.1 "No namespace" instance documents 9
3.2.2 Declarations that define a namespace prefix 10
3.2.3 Default namespace declarations 11

4 URIS AS NAMESPACE NAMES 11

4.1 URI: a locator or just a name? 12
4.2 URI as an XML namespace name 13

5 SCHEMAS AND NAMESPACES 14

5.1 Namespace "visibility" in instance documents 14
5.2 Controlling namespace "visibility" with the schema 16

6 GUIDELINES 16

6.1 URI namespace names 16
6.2 elementFormDefault/attributeFormDefault schema attributes 17
6.3 Namespace defaulting 17
6.4 Minimizing XML namespace complexity in instance documents 17
6.5 NENA XML Artifacts Repository 17

7 REFERENCES 18

TABLE OF FIGURES

Figure 1 - Sample folders 8
Figure 2 - Embedded folder 8
Figure 3 - No namespace instance document 10
Figure 4 - Instance document with namespace declaration 10
Figure 5 - Fully qualified schema instance document 11
Figure 6 - Schema instance document using namespace defaulting 11
Figure 7 - HTTP and URN as XML namespace names 13
Figure 8 - Target namespace declaration 14
Figure 9 - Instance document with mixed namespaces 15

1 Executive Overview

1.1 Purpose and Scope of Document

This document dedicated to namespaces as used in XML technologies.

It is intended to provide a non technical introduction to the concept of namespace and a relatively complete overview of the specific characteristics of XML namespaces.

This document should not be construed as a stand-in for an eventual XML developer's guide. A document that would include NENA recommended best-practices for XML artifact development such as schemas and transforms. This document does however contribute guidelines that may encourage the future development of such a guide.

1.2 Reason for Issue

Many of the more recent NENA standards involve the creation of XML artifacts such as schemas[1] (*schemas*) and WSDL documents. In most cases, these artifacts are considered prescriptive parts of the standards.

Most of these schemas introduce specific XML namespaces for the markup vocabulary they define. XML namespaces play a key role in schema modularity and reuse, both being important requirements in today's information sharing and exchange standards.

Nonetheless, some aspects of XML namespaces can be difficult to grasp. This document attempts to clarify most of those aspects.

1.3 Reason for Reissue

This is the first issue.

NENA reserves the right to modify this document. Whenever it is reissued, the reasons will be provided in this paragraph.

1.4 Recommendation for Standards Development work

Standards development work involving the creation of XML artifacts and associated XML namespaces should follow NENA issued best practices regarding XML namespaces, especially regarding the use of Uniform Resource Identifiers (URI) as namespace names.

This document includes some guidelines for a set of NENA best practices regarding XML namespaces as part of an eventual XML developer's guide.

[1] The term "schema" as used in this document refers to document definitions written in the W3C XML Schema language.

1.5 Costs Factors

Not Applicable.

1.6 Acronyms/Abbreviations

This is not a glossary! See NENA 01-002 - NENA Master Glossary of 9-1-1 Terminology located on the NENA web site for a complete listing of terms used in NENA documents.

The following Acronyms are used in this document:	
AQS	NENA ALI Query Service standard
URI	Uniform Resource Identifier
URL	Uniform Resource Locator (location sensitive)
URN	Uniform Resource Name (location insensitive)
W3C	World Wide Web Consortium
WSDL	Web Service Definition Language
XML	Extensible Markup Language
XSD	W3C XML Schema Definition

1.7 Intellectual Property Rights Policy

1.7.1 General Policy Statement

NENA takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights.

NENA invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard.

Please address the information to:

National Emergency Number Association
4350 N Fairfax Dr, Suite 750
Arlington, VA 22203-1695
800-332-3911
or: techdoccomments@nena.org

2 Introduction to the concept of *namespace*

This section is targeted at to those who are new to the concept of namespace - the specifics of XML namespaces are introduced later in Section 3.

2.1 General concept

Let's attempt a generic definition for our purpose: a ***namespace*** establishes a ***context*** for the non-ambiguous designation of "objects" (concepts, things, persons etc) i.e. within such context, objects have unique identities. Don't forget, namespaces are concerned with object identities, not the objects themselves; so when people say "the object O exists in namespace A" they are really saying "this object has an identity O that belongs to a context established by namespace A".

Here are examples of familiar namespaces (goes to show you the concept has been around for a while):

- The ISBN namespace provides for non-ambiguous (unique) designations of books[2]

  For example, 0764588451 uniquely identifies the book *XML for Dummies by Dykes & Tittel*. in the ISBN namespace.

  The same book also has a unique (non-ambiguous) identity in the Library of Congress Catalog (LCCN) namespace: 2005923240

- Periodic Table of Elements namespace provides for a non-ambiguous (unique) designation of the chemical elements. For example *Xe* for Xenon.

In some cases, we want a short-form lexical notation for associating object identities with their associated namespace. For example instead of "International Standard Book Number identity 0764588451" we would rather write ""ISBN 0764588451". As used here, "ISBN" is known as a ***namespace qualifier*** because it qualifies the object identity in terms of its parent namespace. In the more specific case of XML namespaces, namespace qualifiers are called *namespace prefixes*.

So in the above examples *ISBN 0764588451* and *LCCN:2005923240* are qualified book identities. Apart from allowing short lexical forms, namespace qualifiers help in distinguishing different objects that have the same unqualified identity (both of these advantages are put to good use in XML namespaces as we will see later). For example, it is more likely than not that the books designated as *ISBN 1223334444* and *LCCN:1223334444* are different books. Irrespectively, the point is that the unqualified identity "1223334444" may be ambiguous and the using a namespace qualifier gets rid of the potential ambiguity.

2.2 Moving on to XML

Now let's discuss the concepts introduced above from a perspective that brings us somewhat closer to the usefulness of namespaces as used in XML.

[2] For the curious, take a look at the DOI namespace (Digital Object Identifier)

Consider the two folders in the following figure:

Figure 1 - Sample folders

When referring to file *A* by its file name only, it is unclear which file we are designating; the one in folder *FA* or the one in folder *FB*. However referring to file *A* using a file path like *FA/A*, the name of the folder plays the role of a namespace qualifier: it establishes a non-ambiguous reference context for the files contained in the folder. So we can view a folder as defining a namespace for the files it contains and the folder name as the namespace qualifier to be used when referencing the file from the outside of the folder.

Now assume we want to host the files of folder *FB* under folder *FA*. We cannot simply copy the files of *FB* under *FA* because file names will collide (causing identity ambiguity). We also don't want to change the names of the conflicting source files because other files under *FB* may contain "by name" links (cross-references) to them.

The safest way to proceed then is to copy the folder together with the files it contains (Figure 2). The presence of the folder preserves the reference context for the files it contains.

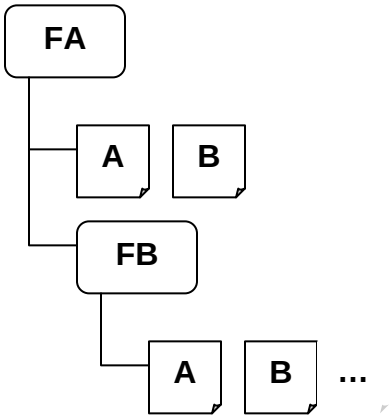

Figure 2 - Embedded folder

This is very similar to how namespaces are used in XML when elements of an XML document are embedded into the structure of another. We use namespace qualifiers to retain the source identity of the "imported" elements, with the additional benefit of preventing names clashes. The former is the real benefit of using namespaces in XML, with the latter being more of a side-effect if of course, the namespace-qualified identities are universally unique – something we will look at in the next section.

Side note – Beyond their ubiquitous use in XML, namespaces are broadly used by the web community for function names, tokens, and identifiers for ever more purposes. For example the NENA XML ALI Query Service (AQS) defines status code values as URIs.

3 XML Namespaces

This section discusses specific aspects of XML. It is not intended as an exhaustive coverage of the topic – to this end you can take a look at the *W3C Namespaces in XML* recommendation hereafter known as the *Namespace Recommendation*. There are also numerous references available from the public Internet.

3.1 Motivation for using namespaces in XML

What follows is an excerpt from the Namespace Recommendation that does a good job of summarizing the rational for using namespaces in XML:

> "*A single XML document may contain elements and attributes (here referred to as a "**markup vocabulary**") that are defined for and used by multiple software modules. One motivation for this is modularity; if such a markup vocabulary exists which is well-understood and for which there is useful software available, it is better to re-use this markup rather than re-invent it.*
>
> *Such documents, containing multiple markup vocabularies, pose problems of recognition and collision. Software modules need to be able to recognize the tags and attributes which they are designed to process, even in the face of "collisions" occurring when markup intended for some other software package uses the same element type or attribute name.*
>
> *These considerations require that **document constructs should have universal names, whose scope extends beyond their containing document**.*"

The Namespace Recommendation goes on to describe a simple method for qualifying element and attribute names (markup vocabulary) with namespace names built out of URI references.

3.2 Namespace declarations

This section introduces the two forms of XML namespace declarations as used in XML instance documents (including XML schemas).

3.2.1 "No namespace" instance documents

There is no "no namespace" declaration as such, but the case nevertheless warrants mention.

As we will see later, unless a schema explicitly specifies to which namespace the markup vocabulary it defines belongs to, the vocabulary belongs to "no namespace". Since valid instance documents of such schemas bear no namespace declaration, its elements are never namespace-qualified.

The following instance document's markup belongs to "no namespace".

```
<?xml version="1.0" encoding="UTF-8"?>
<contact>
  <name>John Doe</name/>
  <phone>1112223333</phone>
  <email>jdoe@home.net</email>
</contact>
```

Figure 3 - No namespace instance document

3.2.2 Declarations that define a namespace prefix

A namespace is usually declared as a user specified attribute, qualified by the XML reserved namespace qualifier ***xmlns*** as follows:

```
xmlns:attributeName="namespaceName"
```

The *attributeName* in the above declaration thereafter can act as a ***namespace prefix***, a short-form synonym for the ***namespace name*** used as the namespace qualifier for the declared namespace. This is why using such a declaration is often known as "declaring a namespace prefix".

Namespace declarations can be attached to any element of a document; not only to the root element of a document.

Here is the a different version of the previous instance document (Figure 3) whose markup vocabulary is declared to belong to the "urn:myspace" namespace. The declared namespace prefix is "*ms*" is used to namespace qualify the root of the document (the enclosed markup is implicitly qualified by the parent namespace).

```
<?xml version="1.0" encoding="UTF-8"?>
<ms:contact xmlns:ms="urn:myspace">
  <name>John Doe</name/>
  <phone>1112223333</phone>
  <email>jdoe@home.net</email>
</ms:contact>
```

Figure 4 - Instance document with namespace declaration

The next example is a schema instance document i.e. one that uses XML Schema namespace defined markup. The instance document of Figure 3 is a valid instance of this schema. You will notice that in this case all document elements are namespace-qualified by the "*xs*" namespace prefix defined as part of the namespace declaration:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="contact">
    <xs:sequence>
      <xs:element name="name"/>
      <xs:element name="phone"/>
      <xs:element name="email"/>
    </xs:sequence>
  </xs:element/>
</xs:schema>
```

Figure 5 - Fully qualified schema instance document

3.2.3 Default namespace declarations

The declaration of a ***default namespace*** uses the XML reserved attribute *xmlns* as follows:

xmlns="namespaceName"

Default namespaces are use in instance documents as one form of control over namespace prefix visibility. For example, here is how the schema of Figure 5 reads when namespace defaulting is used:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="contact">
    <sequence>
      <element name="name"/>
      <element name="phone"/>
      <element name="email"/>
    </sequence>
  </element/>
</schema>
```

Figure 6 - Schema instance document using namespace defaulting

The element to which is attached the default namespace declaration and the entire enclosed markup is implicitly considered part of the namespace (unless otherwise overridden by other enclosed namespace declarations).

4 URIs as namespace names

In this section we take a look at Uniform Resource Identifiers (URI [10]) and their use as namespace names – more specifically XML namespace names.

A URI is a compact sequence of characters that provides a simple and extensible means for identifying an abstract or physical resource.

Each URI begins with a *scheme* name that refers to a specification for assigning identifiers within that scheme [12]. As such, each scheme's specification establishes the syntax and semantics of identifiers using that scheme.

The following URI examples illustrate several URI schemes (in **bold**) and variations in their common syntax components (for our immediate purpose of XML namespace names, we will only consider the *http* and *urn* URI schemes):

ftp://ftp.is.co.za/rfc/rfc1808.txt

http://www.ietf.org/rfc/rfc2396.txt

ldap://[2001:db8::7]/c=GB?objectClass?one

mailto:John.Doe@example.com

news:comp.infosystems.www.servers.unix

tel:+1-816-555-1212

telnet://192.0.2.16:80/

urn:oasis:names:specification:docbook:dtd:xml:4.1.2

4.1 URI: a locator or just a name?

A URI can be further classified as a *locator*, a *name*, or both.

The term *Uniform Resource Locator* (URL [8]) refers to the subset of URIs that, in addition to identifying a resource, provides a means of locating the resource by describing its primary access mechanism (e.g., its network "location"). Because an “http” URI is usually used as an URL, using it as an XML namespace name often confuses users in assuming that there is “something” at the other end of the namespace URL. A namespace URI is simply an identifier. It is not guaranteed to point to anything and, in general, it is a bad idea to assume that it does.

The term *Uniform Resource Name* (URN [9]) has been used historically to refer to both URIs under the "urn" scheme, which are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable, and to any other URI with the properties of a name.

An individual scheme does not have to be classified as being just one of "name" or "locator". Instances of URIs from any given scheme may have the characteristics of names or locators or both, often depending on the persistence and care in the assignment of identifiers by the naming authority, rather than on any quality of the scheme.

The URI syntax is organized hierarchically, with components listed in order of decreasing significance from left to right. For some URI schemes, the visible hierarchy is limited to the scheme itself: everything after the scheme component delimiter (":") is considered opaque to URI processing. Other URI schemes make the hierarchy explicit and visible to generic parsing algorithms.

4.2 URI as an XML namespace name

It is now time to recall the characteristics we would expect of an XML namespace name:

- Unique (to solve markup vocabulary ambiguities)
- Conveys *ownership* (who has defined the namespace)
- Convey *subject path* (what the namespace is about)

For our immediate purpose of XML namespace names, we will only consider the `http` and `urn` URI schemes and briefly look at the merits of each as XML namespace names. Figure 7 uses NENA defined namespace names[3] as samples of these two schemes to illustrate how the syntax of each scheme conveys ownership and subject path information.

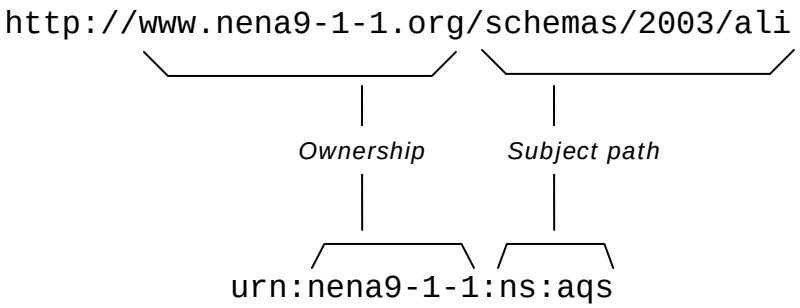

Figure 7 - HTTP and URN as XML namespace names

The requirement that a namespace URI designate a single XML namespace implies that its lexical form be unique. Now, irrespectively of the uniqueness scope of the namespace URI (enterprise, organization, Internet…), it is usually by social convention that this uniqueness can be guaranteed.

Uniqueness of the above URIs in the scope of Internet is achieved through registration of the "ownership" component. In the case of the "http" URI, the "ownership" component is usually a registered DNS domain name while in the case of the "urn" URI, it is normally a registered NID i.e. the *Name Identifier* component of the URN syntax (enclosed between the first pair of colons). The social convention at play here is that of delegated authority, like in the case of IANA-maintained registry of URN Namespace Identifiers (NID).

In the scope of an organization (in this case NENA), uniqueness is expected to be achieved through the NENA managed assignment of subject paths (a social convention in itself).

[3] The "http" scheme URI is the namespace URI currently used for *XML ALI*. The "urn" schema URI is the foreseen namespace URI for *XML ALI Query Service* (the current interim URI is `urn:nena-org:cpe:aqs`)

5 Schemas and Namespaces

We saw earlier that a namespace URI uniquely identifies an XML markup vocabulary. In turn, any given markup vocabulary is defined by one or more schemas. The association of a namespace URI to a markup vocabulary must be specified explicitly in the defining schemas.

By default, the markup vocabulary defined by a schema does not have an associated namespace URI. Such an association must be explicitly specified using a *targetNamespace* attribute (on the root element of the schema) - the namespace URI is thereafter referred to the ***target namespace*** of the schema.

The following schema illustrates this (it is a slightly modified version of the schema of Figure 5). The instance document of Figure 4 is in fact a valid instance of this schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema”
           targetNamespace=”urn:myspace”>
  <xs:element name=”concact”>
    <xs:sequence>
      <xs:element name=”name”/>
      <xs:element name=”phone”/>
      <xs:element name=”email”/>
    </xs:sequence>
  </xs:element/>
</xs:schema>
```

Figure 8 - Target namespace declaration

5.1 Namespace “visibility” in instance documents

When you mix markup vocabularies (elements and attributes) from different namespaces in a single instance document, you may whish to have all instance elements and attributes unqualified so that the namespace origin of each is, for all practical purposes, lexically “invisible”. On the other hand, you may whish to have the namespace origin of elements and attributes be easily and unambiguously derivable from lexical local inspection.

You will have guessed that are interest here is “lexical visibility” i.e. one that relates to instance document legibility.

The sample instance document of Figure 9 illustrates how namespace “visibility” can be used in cases of mixed namespaces. The ***ALIBody*** element belongs to the *NENA XML ALI* namespace as indicated by the namespace declaration attached to the element – here the declaration establishes the namespace as the default namespace so that all enclosed elements default to that namespace. This XML ALI information is being carried as the payload of a ***QueryResponse*** element defined as part of the *NENA ALI Query Service* namespace – here the namespace declaration attached to the root of the instance document also declares “aqs” as the namespace qualifier (prefix).

```
<?xml version="1.0" encoding="UTF-8"?>
<aqs:QueryResponse
   xmlns:aqs="urn:nena9-1-1:ns:aqs"
   inResponseTo="XYZ123" version="0.1" ID="ABC987">
  <aqs:Status>
     <aqs:StatusCode>urn:nena-org:dtc:aqs:status:Ok</aqs:StatusCode>
  </aqs:Status>
  <aqs:QueryProperties>
     <aqs:TrunkID>12</aqs:TrunkID>
     <aqs:CallTakerPosition>22</aqs:CallTakerPosition>
  </aqs:QueryProperties>
  <aqs:QueryResultData>
     <ALIBody xmlns="http://www.nena9-1-1.org/schemas/2003/ali"
         schemaVersion="4.2" >
          <CallInfo>
             <CallbackNum>1234567890</CallbackNum>
               <CallingPartyNum>0987654321</CallingPartyNum>
               <ClassOfService code="1">String</ClassOfService>
               <TypeOfService code="0">String</TypeOfService>
               <SourceOfService>W</SourceOfService>
          </CallInfo>
          <LocationInfo>
               <GeoLocation>
                  <Latitude>3.141592</Latitude>
                  <Longitude>3.141592</Longitude>
               </GeoLocation>
          </LocationInfo>
          <Agencies>
               <ESN>00000</ESN>
          </Agencies>
          <SourceInfo>
               <DataProvider>
                  <DataProviderID>Strin</DataProviderID>
               </DataProvider>
               <AccessProvider>
                  <AccessProviderID>Strin</AccessProviderID>
               </AccessProvider>
               <ALIRetrievalGMT>2001-12-17T09:30:47.0Z</ALIRetrievalGMT>
          </SourceInfo>
          <NetworkInfo>
               <PSAPID>Stri</PSAPID>
               <RouterID>Stri</RouterID>
          </NetworkInfo>
       </ALIBody>
  </aqs:QueryResultData>
</aqs:QueryResponse>
```

Figure 9 - Instance document with mixed namespaces

5.2 Controlling namespace "visibility" with the schema

When a schema defines a target namespace, two specialized attributes control namespace visibility in valid instance documents i.e. the usage rules for namespace qualifiers: *elementFormDefault* and *attributeFormDefault*[4].

Each can independently be set to either of the following values:

- *unqualified*

  If set to "*unqualified*", elements that are part of the target namespace need not be namespace-qualified. Consequently you may not always be able to tell which namespace an element belongs to without some knowledge of the schema.

- *qualified*

  If set to "*qualified*", elements that are part of the target namespace must be explicitly namespace-qualified. In other words, they are not implicitly associated with a namespace. Consequently, you will always be able to tell to which namespace an element belongs by local inspection of the instance document.

Now attribute names can be uniquely identified based on the element type to which they belong, so there is no need identify them further by including them in an XML namespace i.e. they are implicitly associated with the namespace of the element to which they are attached. In fact, the only reason for allowing attribute names to be namespace-qualified is so that attributes defined in a given namespace can be used with elements defined in another namespace.

6 Guidelines

Each of the sub-sections below offers guidelines related to XML namespaces used during NENA standards development. These are stated rather briefly and are intended as seeds for a set of NENA recommendations that could be introduced in a future NENA publication.

6.1 URI namespace names

The "urn" scheme URI is preferred over the "http" scheme.

However, it may be impractical to change existing schemas after the fact. NENA should however enforce a standard URI "ownership" component syntax across all NENA schemas for both "urn" and "http" URI schemes as follows:

- `urn:nena9-1-1:...`
- `http://www.nena9-1-1.org/...`

[4] These attributes also affect what namespace information validating parsers attach to internal representations of valid instance documents, but this subject is beyond the scope of this document.

Some standardization of URI "subject path" syntaxes may also be considered:

- if desired, the organizational identity of the originating entity should appear at the beginning of the path; for example the acronym of the NENA TC.
- Using "ns" for XML namespace URIs

6.2 elementForDefault/attributeFormDefault schema attributes

Use the following combination:

- *elementFormDefault* should be set to the value "qualified"
- *attributeFormDefault* should be set to the value "unqualified" (the default value)

6.3 Namespace defaulting

This is purely a matter of choice and affects only readability of the document. It does not change the namespace information validating parsers attach to the internal representation of instance documents.

6.4 Minimizing XML namespace complexity in instance documents

You can minimize this complexity by doing the following whenever appropriate:

- Declare all XML namespaces on the root element
- Use one prefix per XML namespace

  More than one is allowed and can be useful in certain circumstances

6.5 NENA XML Artifacts Repository

When a NENA standard involves the creation of XML artifacts such as schemas (*schemas*), WSDL documents and associated namespace URIs, these artifacts effectively become prescriptive parts of the standard.

Just as standard texts are made publicly accessible, so should the associated XML artifacts, thus the need for NENA to develop an *XML (Artifact) Repository* accessible through its web site.

The artifact repository is the "place" where the artifacts (files) are physically kept.

The repository characteristics should minimally include:

- Web-site interface (browser navigation)
- Retrieval by individual artifact
- Retrieval by *artifact family* (archive of all artifacts associated with a given namespace URI)
- Access to older versions of artifacts or artifact sets

Navigating the repository may or may not follow the namespace hierarchy.

An interface or mechanism for program access to repository resources is not mandatory.

An XML namespace may have associated documentation (specifications, reference material, tutorials, etc., perhaps in several formats and several languages), schemas (in any of several forms), style sheets, software libraries, applications, or any other kind of related resource. A user, encountering a namespace might want to find any or all of these related resources. The XML Repository should inter-link all these.

7 References

[1] W3C - Namespaces in XML
[2] XML Namespace Tutorial
[3] W3C – XML 1.0 Specification
[4] W3C – Architecture of the World-Wide-Web, Vol.1
[5] W3C – XML Schema
[6] IANA - URN Namespaces Registry
[7] IETF – Uniform Resource Names (URN) Namespace Definition Mechanisms (RFC 3406)
[8] IETF – Uniform Resource Locators (URL) (RFC 1738)
[9] IETF – URN Syntax (RFC 2141)
[10] IETF – Uniform Resource Identifier (URI) – Generic Syntax (RFC 3986)
[11] IETF – Internationalized Resource Identifiers (IRI) (RFC 3987)
[12] IANA – Registry of URI Schemes (RFC 4395)