XML Namespaces & Origo Messages

Policy Document

Version: ***2.0***

Date: ***31/01/2013***

Distribution: ***Industry***

CHANGE HISTORY

Date	Version	Change(s) Incorporated
21st January 2004	1.0	Initial version documenting the Namespace Policies between: • 1998-2002 (no namespace) • 2003-2005 (changed to apply a unique namespace per business domain e.g. Quote and New Business)
31st January 2013	2.0	Updated version documenting the Namespace Policies for the periods:- • 1998-2002 (no namespace) • 2003-2005 (changed to apply a unique namespace per business domain e.g. Quote and New Business) • 2006- present (changed to apply a unique namespace per schema)

TABLE OF CONTENTS

1 Introduction

1.1 Namespace Policies

This document describes the policies adopted by the Origo Standards holder's community for the use of XML Namespaces within the Origo Standards schemas and associated XML messages.

Due to the nature of Standards development and the different versions supported there are actually three namespace policies which are still in use. Which policy is applicable is dependent on which versions of the Origo Standards are in use by particular implementers. A matrix relating each Origo Standard to the relevant XML Namespace policy is provided in Section 3.

The three namespace policies in existence are as follows:

Date	Namespace Policy
1998-2002	In 1998 Origo produced its first set of Standards and adopted a policy where no namespace was used. The QNB Standards still comply with this policy. With little XML experience amongst implementers in the Life & Pensions Industry it was agreed that Origo Standards would initially not make use the namespace feature of XML Schema.
2003-2005	In 2003 Origo introduced namespaces at the Process Domain level (by process domain we mean for example, QNB or Contract Enquiry etc.). This policy was introduced after implementers experiences with XML matured and namespaces became a manageable concept.
2006-present	In 2006 Origo introduced namespaces at the individual schema level. This policy was introduced due to the way in which SOAP based web service frameworks were implemented, thus requiring namespaces applicable to individual schemas.

More details for each namespace policy, why it was adopted and what it involves are included later in this document.

1.2 What is an XML Namespace?

In simple terms, a Namespace in XML provides a mechanism for two XML information items (elements or attributes) that share the same name, but have different meanings/contexts and to be referred to unambiguously.

They were introduced to enable users to mix XML vocabularies that might potentially use the same names for XML components. By binding elements to different Namespaces their distinct meaning can be simply and effectively preserved by processing applications, in the same way that programming languages use Namespaces to enable two or more classes with the same name to be used in the same system.

A more detailed explanation of XML Namespaces is provided in Appendix A of this document.

2 Origo XML Namespace Policies

The History of XML Namespaces Used in Origo Messages

2.1 1998-2002

Origo first started to develop XML message structures in 1998, with the first version of the Quotes Standard. This Standard was finalised at roughly the same time that the W3C finalised the Namespaces specification. Therefore, it was not appropriate to implement Namespaces at that time.

2.1.1 What was Namespaced?

Because Namespaces were a new concept it was not appropriate to implement Namespaces at that time (1998). At that point Origo adopted the view that because Origo messages are used in a closed community, there was not a great need to complicate the messages with Namespaces. When Origo started to produce XML Schema in the early part of 2001, we did define a single Namespace (http://www.origoservices.com), but the only reason for introducing this was to make the schema valid in terms of the W3C XML Schema specification.

It was also our understanding that although there were implementers who used Namespaces properly, in general most did not define Namespaces in their message instances, and if a Namespace was included by the sender, the receiver tended to ignore this.

2.2 2003-2005

The approach Origo had taken up till 2003 with XML Namespaces could in no way be described as "best practice", and as a result this was changed. The following major issues existed and formed a compelling case for change.

Non-Namespace aware Software

At the time it was possible to write software to process Origo messages which were not Namespace aware. However, as our requirements become more sophisticated, we started to augment Origo designed structures with components that were designed and maintained by other standards bodies (for example W3C XML Signatures), or we needed to use constructs which relied on particular XML functionality that had to be implemented with Namespaces (e.g. xsi:type). There was a risk that non-Namespace aware software would not be able to handle these parts of a message. A concrete example of this came to light early in 2003 when it was discovered that software, that was widely used within the industry, could not produce a New Business message that conformed to the Standard, due to its inability to handle the Namespaces, for the xsi:type feature, used within the underwriting questions structure to choose between a Standard or non-Standard question code.

Combining Messages

In 2003 two of our Standards (Remote Publishing and Tracking) had requirements to place a message of another type within another message (such as returning the results of an extranet link for a quote, the quote response is placed inside the Remote Publishing Alert message). If each different Origo message type had a different Namespace, combining the different message types would be made simpler and would allow useful things such as schema validation to be performed against the combined message.

Extensibility/Interoperability

In 2003 Origo were expecting the emerging SOAP message protocol to become key to messaging in the very near future. SOAP mandates that XML sent over it must be Namespaced. Origo expected that most implementers would purchase a complete SOAP platform or tool kit from one of the many software suppliers who had such products available. To configure a single SOAP platform to send/receive messages of more than one type (e.g. to process Quotes and Contract Enquiry messages using the same SOAP box), Origo expected that most SOAP platforms would need each message type to be in its own Namespace, to enable the platform to execute the correct set of code to process each message.

Where it was appropriate Origo designed into the message structures **tpsdata** elements, as placeholders for implementers to extend the standard Origo messages with their own data elements. Again, this was not what most experts consider to be best practice, XML (as its name suggests) is supposed to be extensible. Using a separate Namespace should enable implementers to extend a standard message in any way they desire (what Namespaces were intended to be used for). This could simplify things considerably for message receivers, enabling them to have a clean separation of the code that was created to handle each Namespace. They could also more easily filter out and ignore anything from a specific Namespace that they did not wish to process. Several implementers had already expressed the opinion that Origo should remove specific **tpsdata** elements and use XML in the way it was designed.

In the early days, when XML was new and poorly understood, there was a need to keep things simple to aid implementation. By 2003 we had more implementers within the Origo community, who had a better understanding of XML (which is now very clearly a mainstream technology that is supported heavily by good quality software tools). Origo's use of XML was looking to become more sophisticated, following established best practice and full conformance with (IT) industry standards.

Due to the impact of introducing/changing Namespaces, Origo proposed to phase in new Namespaces, applying them to Standards when an appropriate new version of the Standard was released, when changes to the Standard would force implementers to update their code/XSLT etc.

2.2.1 What was Namespaced?

In 2003 Origo introduced the concept of a Namespace for each of its “Process Domains”, e.g. Quotes and New Business would exist in a single Namespace, Contract Enquiry would have a separate Namespace, as would Commission and all the other Origo process based Standards.

Some components were common to more than one Standard, such as the Origo Message Header (as defined by the HTTP Message Transmission Guidelines). These were also placed in their own Namespaces. This simplified the incorporation of the schemas for these components into the schemas for other Namespaces, for both Origo and those Implementers who have used these components directly within their own XML environment.

2.2.2 What was the scope of each Namespace?

A Namespace was to be created for each “Major Version” of a Process Domain. This covered all the different types of messages that could be created for that process domain. For example Quotes and New Business v4 would be represented by a single Namespace, containing all Requests and Responses for all the different product and sub-product type messages that could be produced.

Origo minor standard versions should conform to the following rules:-

1. Semantics of elements and attributes would not be altered;
2. Changes to the document involved the addition of elements and attributes, but not removal.

These rules made the Origo schemas backward compatible. Therefore, all minor versions of a Standard would exist within the same Namespace, with the Namespace for the Standard only requiring to be changed for a new major version release.

This decision was reached after considering the opinions of other XML experts on this point. There was general consensus that a Namespace should be changed as infrequently as possible, as this could impact code and XSLT written against the Namespace.

Implementers would require change to code/XSLT to implement the new additions to a Standard that come with a minor version change. However, if implementers designed their code to only process XML elements/attributes that they are interested in, ignoring the rest, it would be possible to write an application that will still work with any future minor releases of a Standard in that Namespace.

2.2.3 Changes to message appearance

As best practice, Origo believes that implementers should always Namespace prefix ALL XML Elements and Attributes within an instance document/message. This ensures that problems are not encountered with issues relating to “default namespaces” and their scope on attributes (i.e. if you do not declare a Namespace prefix on an attribute it does not belong to a Namespace).

NOTE: there is one exception to this rule, which is detailed in the Conformance section of the XML Namespaces Specification[1]. Any XML Attribute that is defined as type "ID" or "IDREF", within the schema, should NOT be Namespaced.

Below is an example of how XML would look when conforming to the Origo XML Namespace Policy of 2003-2005 with explicit prefixing:-

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mtg:message xmlns:mtg="http://www.origostandards.com/schema/mtg/v2">
    <mtg:m_control id="m_control">
        <mtg:control_timestamp>2003-11-12T00:00:00</mtg:control_timestamp>
        <mtg:message_id>C0CB3A4D-2DD8-47E8-8149-2F5B5AD788E4</mtg:message_id>
    </mtg:m_control>
    <qnb:m_content xmlns:qnb="http://www.origostandards.com/schema/qnb/v4"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <qnb:intermediary qnb:type="IFA"/>
        <qnb:application>
            <qnb:product/>
        </qnb:application>
    </qnb:m_content>
</mtg:message>
```

2.2.4 Namespace Format Conventions

Origo adopted the following conventions for namespace naming:-

1. The Namespace URI would be composed of 3 parts:-

Root	Standard	Version

http://www.origostandards.com/schema
 	
/schema is used to enable Origo to locate all Namespace definitions on our website in the same location.
	/qnb
	
	A short abbreviation of the Standards Name, as used at the beginning of MIG and Schema Documents.
	qnb = Quotes and New Business
	ce = Contract Enquiry
	mtg = Message Transmission Guidelines
	rp = Remote Publishing
		/v4
		
		The major version number of the Standard to which this Namespace URI applies, prefixed by "v"

E.g. http://www.origostandards.com/schema/qnb/v4 Quotes and New Business version 4

2. The Namespace Prefix will be the same as the "Standard" component of the Namespace URI (see above). For example <qnb:application> (**qnb** = Quotes and New Business). The exception to this is for the Origo Data Types library which will be prefixed "**origo**" (to minimise changes required to existing schema, which already follow this convention).

[1] http://www.w3.org/TR/REC-xml-names/#Conformance

3. It is Origo's intention to publish an RDDL on the Origo Website (in an unsecured area), at each location that is to be used as a Namespace, to enable the URL placed in instance documents/messages for the Namespace, to resolve properly on the web. This file will give a brief description of the Standard and will point to where additional information can be found (within the secure area of the website).

2.2.5 Implementation Timetable

As stated earlier the aim when implementing Namespaces was to minimise the impact of the change. To this end Origo adopted the implementation plan in June 2003 of introducing Namespaces to all Standards when there was sufficient reason to make a major change to each Standard. The timing of such changes would be agreed by the Governance Groups.

2.2.6 Impact of changing Namespaces

Introducing Namespaces or changing the way an XML vocabulary is namespaced is not a trivial exercise. By changing the Namespace, the context of each data item within that vocabulary changes (in effect re-naming every element and attribute). This will mean that there could be an impact on every line of XML related code that is implemented for that vocabulary.

2.3 2006-Present

The approach Origo had taken between 2003 and 2005 with XML Namespaces was to apply namespaces at the Process Domain level (e.g. Quotes and New Business). At the time it was thought that this would be sufficient to cope with the requirements of the emerging SOAP message protocol and the use of SOAP platforms and tool kits. Unfortunately this was not the case.

SOAP tool kits have the capability to generate skeleton application code from the schemas which represent the messaging structures. This is one of the major advantages when choosing to use a SOAP tool kit. The complex code required to marshal data to/from an XML message and a proprietary back end software implementation is generated by the toolkit, thus saving considerable time and effort.

Unfortunately during 2005 it was discovered by a few implementers that the Origo XML Namespace policy was causing code generation issues. Because the majority of code generation tools use the target namespace from the schema to construct class package names, it was necessary to ensure that both the request and response schemas had different target namespaces (thus avoiding class name clashes). This was not compatible with the Origo namespace policy of 2003-2005 where the same namespace was used across all the schemas in a process domain (request and response messages).

This could only be corrected in one of two ways:-

1. The SOAP tool kits had to be changed to accommodate the contemporary namespace policy
   OR
2. The Origo namespace policy had to change.

It seemed unlikely that the Origo Standards holder's community would be able to persuade a large number of software vendors to change their products as required, so the decision was taken by the Origo Technical Governance Group that the namespace policy should be changed to enforce a unique namespace per schema.

2.3.1 What was Namespaced?

In 2006 Origo introduced the current Namespace policy which required that each of its schemas used a distinct namespace, e.g. every Origo Standards process domain, product, product sub type, request and response schemas would have a unique Namespace.

2.3.2 What was the scope of each Namespace?

A Namespace was to be created for each Origo schema. When a Standard was moved to new major version, all related schemas would be modified to use a new namespace.

Once again, as for the previous policy, Origo minor standard versions should conform to the following rules:-

1. Semantics of elements and attributes would not be altered;
2. Changes to the document involved the addition of elements and attributes, but not removal.

These rules make the Origo schemas backward compatible.

2.3.3 Changes to message appearance

As best practice, Origo believes that implementers should always Namespace prefix ALL XML Elements and Attributes within an instance document/message. This ensures that problems are not encountered with issues relating to "default namespaces" and their scope on attributes (i.e. if you do not declare a Namespace prefix on an attribute it does not belong to a Namespace).

NOTE: there is one exception to this rule, which is detailed in the Conformance section of the XML Namespaces Specification[2]. Any XML Attribute that is defined as type "ID" or "IDREF", within the schema, should NOT be Namespaced.

Below is an example of how XML would look when conforming to the Origo XML Namespace Policy of 2006-Present with explicit prefixing:-

[2] http://www.w3.org/TR/REC-xml-names/#Conformance

```
<?xml version="1.0" encoding="UTF-8"?>
<mtg:message xmlns:mtg="http://www.origostandards.com/schema/mtg/v2">
    <mtg:m_control id="m_control">
        <mtg:control_timestamp>2006-11-12T00:00:00</mtg:control_timestamp>
        <mtg:message_id>C0CB3A4D-2DD8-47E8-8149-2F5B5AD788E4</mtg:message_id>
    </mtg:m_control>
    <qbondsreq:m_content
                        xmlns:qbondsreq="http://www.origostandards.com/schema/qbonds/v4/req"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        < qbondsreq:intermediary qbondsreq:type="IFA"/>
        < qbondsreq:application>
            < qbondsreq:product/>
        </ qbondsreq:application>
    </ qbondsreq:m_content>
</mtg:message>
```

2.3.4 Namespace Format Conventions

Origo adopted the following conventions for namespace naming:-

1. The Namespace URI would be composed of 4 parts:-

Root	Standard	Version	Message Type
http://www.origostandards.com/schema /schema is used to enable Origo to locate all Namespace definitions on our website in the same location.	/qbonds A short abbreviation of the Standards Name, as used at the beginning of MIG and Schema Documents. qbonds = Quotes Bonds	/v4 The major version number of the Standard to which this Namespace URI applies, prefixed by "v"	/req The message type makes the namespace unique. Values could be req (request), resp (response), ack (acknowledgment)

E.g. http://www.origostandards.com/schema/qbonds/v4/req Quotes Bonds version 4 Request

2. The Namespace Prefix will be the same as the "Standard"+"Message Type" components of the Namespace URI (see above). For example <qbondsreq:application> (**qbondsreq** = Quotes Bonds Request).

3. It is Origo's intention to publish an RDDL file on the Origo Website (in an unsecured area), at each location that is to be used as a Namespace, to enable the URL placed in instance documents/messages for the Namespace, to resolve properly on the web. This file will give a brief description of the Standard and will point to where additional information can be found (within the secure area of the website).

2.3.5 Implementation Timetable

As stated earlier the aim when implementing Namespaces was to minimise the impact of the change. To this end Origo adopted the implementation plan in 2006 of introducing Namespaces to all Standards when there was sufficient reason to make a major change to each Standard. The timing of such changes would be agreed by the Governance Groups.

2.3.6 Impact of changing Namespaces

Introducing Namespaces or changing the way an XML vocabulary is namespaced is not a trivial exercise. By changing the Namespace, the context of each data item within that vocabulary changes (in effect re-naming every element and attribute). This will mean that there could be an impact on every line of XML related code that is implemented for that vocabulary.

3 Namespace Policy / Origo Standard Matrix

With the existence of three namespace policies (depending on which version of an Origo Standard you are interested in) it would be useful to know which applies to each Origo Standard. Below is table containing the Standards, the supported versions and the namespace policy they adhere to. Note that all future Origo Standards should comply with the “2006-present” policy.

Standard	Version	Namespace Policy Adhered To
Adviser	1.0	2006-present
Agency Code Population	1.0	1998-2002
Business Contacts	1.0	2006-present
	1.1	2006-present
Cash Flow	1.0	2006-present
	1.1	2006-present
Commission (EDIFACT)		Not using XML
Commission Transfer (XML)	1.0	1998-2002
Contract Enquiry Bonds Valuation	2.0	MTG 2003-2005 SOAP 2006-present
	2.1	2006-present
Contract Enquiry Collective Investments Valuation	1.2.1	MTG 1998-2002 SOAP 2006-present
	1.3	2006-present
Contract Enquiry Endowment Single & Multi Valuation	2.0	MTG 2003-2005 SOAP 2006-present
	2.1	2006-present
Contract Enquiry Pensions Multi Contract Valuation	2.2	2006-present
Contract Enquiry Pensions Single & Multi Contract Valuation	2.0	MTG 2003-2005 SOAP 2006-present
Contract Enquiry Pensions Single Contract Valuation	2.1	2006-present
	2.2	2006-present
Contract Enquiry Transaction History Bond	2.0	2003-2005
	2.1	2006-present
	2.2	2006-present
Contract Enquiry Transaction History Endowment	2.0	2003-2005
	2.1	2006-present
	2.2	2006-present
Contract Enquiry Transaction History Pension	2.0	2003-2005
	2.1	2006-present
	2.2	2006-present
Contract Enquiry Transaction History WOL Protection	2.0	2003-2005
	2.1	2006-present
	2.2	2006-present
Contract Enquiry Whole Of Life Single & Multi Valuation	2.0	2006-present
	2.1	2006-present
Contract Enquiry Wrap Valuation	1.0	2006-present
Contracts Summary	1.0	2006-present
Contribution Collection	1.0	2003-2005
Employment	1.0	2006-present
Extranet Linking	1.0	No schemas
	1.1	2003-2005

Flexible Integration Toolkit (pre-pop)		See individual Pre-pop patterns
Generic Data	1.0	2006-present
	2.0	2006-present
HTTP Message Transmission	1.3	2003-2005
	2.0	2006-present
	2.1	2006-present
Investment Strategy	1.0	2006-present
	1.1	2006-present
Legal Framework		No schemas
Maintain Adviser	1.0	2006-present
Maintain Business Contacts	1.0	2006-present
	1.1	2006-present
Maintain Cash Flow	1.0	2006-present
	1.1	2006-present
Maintain Contract Summary Details	1.0	2006-present
Maintain Contracts Summary	1.0	2006-present
Maintain Employment	1.0	2006-present
Maintain Generic Data	1.0	2006-present
	2.0	2006-present
Maintain Investment Strategy	1.0	2006-present
	1.1	2006-present
Maintain Personal Fact Find	1.0	2006-present
Maintain Portfolio Summary	1.0	2006-present
	1.1	2006-present
Maintain Product Features	1.0	2006-present
	1.1	2006-present
Maintain Property Features	1.0	2006-present
	1.1	2006-present
Money Laundering	1.0	2003-2005
New Business Annuities	2.0	MTG 1998-2002 SOAP 2006-present
New Business Bonds	3.4	1998-2002
	3.5	1998-2002
	3.7	1998-2002
New Business Collective Investments	3.4	1998-2002
	3.5	MTG 1998-2002 SOAP 2006-present
	3.7	MTG 1998-2002 SOAP 2006-present
New Business Endowments	3.4	1998-2002
	3.5	1998-2002
	3.7	1998-2002
New Business Group Pensions	3.4	1998-2002
	3.5	1998-2002
	3.7	MTG 1998-2002 SOAP 2006-present
New Business Individual Pensions	3.4	1998-2002
	3.5	1998-2002
	3.7	1998-2002
New Business Protection	3.4	1998-2002
	3.5	1998-2002
	3.7	1998-2002
New Business SIPP/IDPR	1.0	2006-present

	1.1	2006-present
	1.2	2006-present
New Business Tracking	2.0.1	2003-2005
	2.1	2003-2005
	2.2	2003-2005
Occupation Codes List		No schemas
Personal Fact Find	1.0	2006-present
Portfolio Summary	1.0	2006-present
	1.1	2006-present
Product Features	1.0	2006-present
	1.1	2006-present
Property Features	1.0	2006-present
	1.1	2006-present
Provide Contract Valuation	1.0	2006-present
Quotes Annuities	3.5	1998-2002
	3.6	1998-2002
	3.7	1998-2002
Quotes Bonds	3.4	1998-2002
	3.6	1998-2002
	3.7	1998-2002
Quotes Endowments	3.4	1998-2002
	3.6	1998-2002
	3.7	1998-2002
Quotes Group Pensions	3.4	1998-2002
	3.6	1998-2002
	3.7	MTG 1998-2002 SOAP 2006-present
Quotes Individual Pensions	3.4	1998-2002
	3.6	1998-2002
	3.7	1998-2002
Quotes New Business Domain		No schemas
Quotes Protection	3.4	1998-2002
	3.6	1998-2002
	3.7	1998-2002
Quotes SIPP/IDPR	1.0	2006-present
Receive Automatic Enrolment Employee List	1.0	2006-present
Receive Group Scheme Contribution List	1.0	2006-present
Receive External Alert	1.0	2006-present
	1.1	2006-present
	1.2	2006-present
Remote Publishing	2.0	1998-2002
	3.0	2003-2005
Remuneration Statement (XML)	1.0	2006-present
	1.1	2006-present
Retrieve Documentation	1.0	2006-present
Standard Data Types	1.4.1	1998-2002
	2.0	2003-2005
Switches & Redemptions	1.0	2003-2005

4 Appendix A

XML Namespaces were introduced as a W3C (World Wide Web Consortium) recommendation on 14th January 1999. The Namespace recommendation defines a method for declaring a one or two part Namespace name, and binding it to elements and attributes in an XML instance.

Namespaces for XML were developed to allow users to mix XML vocabularies that might potentially use the same element and attribute names. By binding elements to different Namespaces their distinct meaning can be simply and effectively preserved by processing applications.

For example:-

```
<my_root_element>
<envelope>
    <send_to>
        <server>
            <name>my_server</name>
            <address>some_ip_address_here</address>
        </server>
    </send_to>
</envelope>
<content>
    <an_order_for_something>
        <supplier>
            <name>Frederick The Slightly Bald</name>
            <address>
                <line_1>…</line_1>
                <etc/>
                </address>
            </supplier>
        </an_order_for_something>
</content>
</my_root_element>
```

Ambiguity could arise if application modules need to rely on one of the two uses of “name” and “address”, and this could be significantly magnified if these elements appeared more-or-less anywhere within an XML instance (as with XSLT, or XSL-FO elements).

However, if these elements are associated with different Namespaces then any ambiguity is removed.

```
<my_root_element>
  <env:envelope xmlns:env=”http://www.my_org.com/env”>
    <env:send_to>
      <env:server>
        <env:name>my_server</env:name>
        <env:address>some_ip_address_here</env:address>
      </ env:server>
    </ env:send_to>
  </ env:envelope>
  <content >
    <po:an_order_for_something xmlns:po="http://www.my_org.com/purchase_order">
      <po:supplier>
        <po:name>Frederick The Slightly Bald</po:name>
          <po:address>
                <po:line_1>…</po:line_1>
          <po:etc/>
        </po:address>
      </po:supplier>
    </po:an_order_for_something>
  </content>
</my_root_element>
```

Now “env:name” and “env:address” belong to the namespace http://www.my_org.com/env, while “po:name” and “po:address” belong to the Namespace http://www.my_org.com/purchase_order, and can be clearly distinguished.

4.1 Anatomy of a namespace

To use Namespaces, one first declares one or more Namespaces within an XML instance. Namespaces are bound to XML elements either implicitly, or explicitly; and to attributes explicitly.

A Namespace can consist of two parts, a universal name, and a local prefix. The universal name is expressed as an URI (Uniform Resource Identifier), and is, at the moment, always a URL (Uniform Resource Locator). The URI used should remain the same across many instances of XML documents using the same Namespace.

If declaring a Namespace as a default Namespace, only the URI is required, and the Namespace applies to all elements within scope. Note that a default Namespace only applies to elements, **not** to attributes.

Alternatively a Namespace may be qualified by a prefix (any characters valid as part of an XML element name, as long as it does not begin with the three letters “x”, “m” or “l”). This allows a Namespace to be applied selectively to elements and attributes within scope. The prefix is local to the Namespace declaration, and it is legal to use different prefixes for the same Namespace in different XML instances, and even within one XML instance.

Note that Namespace aware software should only use the local prefix as a link to the universal name. It should not use the prefix itself to disambiguate. It is quite legal for the same Namespace to apply to more than one part of an XML document, and to use a different prefix in each part. Despite the different prefix, elements are still bound to the same Namespace.

For example:

```
<example>
```

```
    <example1 xmlns:eg1="http://www/my/namespace">
        <eg1:fred/>
    </example1>
    …
    <example2 xmlns:eg2="http://www/my/namespace">
        <eg2:bob/>
</example>
```

"eg1:fred" and "eg2:bob" both belong to the namespace http://www/my/namespace.

4.2 Declaring a namespace in an XML instance

Namespaces are declared by use of the *xmlns* attribute within an XML element, in one of two ways:

4.2.1 Default namespace

xmlns="my_URI"

For example:

```
<x xmlns="http://this.is.my.namespace.com/so.there">
    <other_elements_within_my_namespace/>
</x>
```

4.2.2 Prefixed namespace

xmlns:my_prefix="my_URI"

For example:

```
<x xmlns:fred="http://this.is.my.namespace.com/so.there">
    <element_outside_my_namespace fred:my_attr="this attribute is within my namespace"/>
    <fred:element_within_my_namespace my_attr="this attribute is outside my namespace"/>
</x>
```

Note that the prefix is declared by extending the xmlns attribute name. It is valid to declare many Namespaces within the same element (only one of which can be default). It is even valid to declare the same Namespace more than once, as long as you use a different prefix each time.

4.3 Namespace Scope

A Namespace name is in scope for the element in which the Namespace is declared, and all element descendants.

If a default Namespace is declared, the element in which the Namespace is declared, and all descendant elements are automatically within the Namespace, unless it is explicitly overridden by another Namespace declaration. If a default Namespace is declared with an empty URI, this has the effect of there being no default Namespace within the scope of the declaration.

If a prefixed Namespace is declared then all element and attribute descendents have the potential to belong to the Namespace.

4.4 What can we really use namespaces for?

According to the recommendation, Namespaces have just the one function; removing ambiguity when multiple XML vocabularies are combined. So if you are using one vocabulary, in isolation, there is no need to use Namespaces at all. If someone else takes your vocabulary and combines it with another, they can perfectly well declare their own Namespaces as necessary.

4.5 The URI

Uniform Resource Identifiers (URI) are compact strings of characters for identifying an abstract or physical resource.

The important thing to note is that an identifier is not necessarily a locator. URLs are a type of URI, and yes, they do provide a pointer to the location of a resource. However, the Namespace recommendation states that a Namespace only need be a URI, it doesn't have to be a URL. Consequently it should not be assumed that a Namespace name, even when expressed as a URL, actually points to something real. However, people have got used to the idea that a URL points to something, and so have the not unreasonable expectation that URLs used as Namespaces should do the same.

4.6 Namespaces and XML Schema

The XML schema recommendation is very careful to reinforce the role of Namespaces as set out in the Namespace recommendation. XML Schema provides a set of attributes for establishing Namespaces and controlling their application to elements and attributes. An element or attribute bound to a Namespace is said to be qualified.

The schema element can have the following attributes:-

targetNamespace: the URI of the namespace to use in instance documents. The modular nature of schemas means that instance documents may need to declare more than one Namespace to be conformant. All globally declared elements and attributes (i.e. those declared as immediate children of the schema element) must be qualified in an XML instance.

elementFormDefault: has the value "*qualified*" if local elements must belong to the target Namespace, and "*unqualified*" if they must not. A local element is one declared as a descendant of a global element. The default value is "*unqualified*".

attributeFormDefault: has the value "*qualified*" if local attributes must belong to the target Namespace, and "*unqualified*" if they must not. The default value is "*unqualified*".

For example:
A recent Origo XML schema for an Adviser Remuneration Statement takes the following form:-

```
<xsd:schema    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               targetNamespace="http://www.origostandards.com/schema/RemunerationStatement/v1"
               xmlns:remst="http://www.origostandards.com/schema/RemunerationStatement/v1"
               elementFormDefault="qualified"
               attributeFormDefault="qualified">
```

So conformant XML instances must adhere to one of these Namespace declarations:

1. declare this target Namespace explicitly using a Namespace prefix

   ```
   <message xmlns:remst=http://www.origostandards.com/schema/RemunerationStatement/v1
   ```

2. declare this target Namespace implicitly using a default Namespace.

   ```
   <message xmlns=http://www.origostandards.com/schema/RemunerationStatement/v1
   ```

Individual element and attribute declarations may override **elementFormDefault** and **attributeFormDefault** attributes by setting the **form** attribute to the value of "*qualified*" or "*unqualified*".

Elements and attributes, both global and local must be qualified unless the schema definition of a particular element or attribute overrides this (which is typically done in Origo schemas for ID/IDREF attributes).

4.7 XML Schema Data Types

XML Schema includes two data types pertaining to Namespaces:

Qname

Qname = Qualified name. This is an element name qualified by a Namespace prefix. Clearly this can only be used in scope of a Namespace declaration.

NCName
NCName = non-colonised name. A string which can be a legal XML element name, with the added restriction that it cannot contain a colon. This guarantees that the name can be legally used with Namespaces.

4.8 Namespaces in XSLT

XSLT is a fully Namespace aware application. This means that Namespaces can be navigated in the source tree, and added to result trees. Namespace forms one of the XPath navigation axes, so one can build an expression which filters out elements and attributes belonging to particular Namespaces, which is after all the whole point of Namespaces. If you need to build an expression which navigates to an element belonging to a Namespace, then your query must reflect this. XSLT compares Namespace names as strings, so all characters in two Namespaces must be completely identical to be considered to be the same (i.e. the comparison is case sensitive).

XSLT uses Namespaces itself, to reserve element names which form part of the XSLT vocabulary. Indeed XSLT is a good example of how useful Namespaces can be to processing software. An XSLT processor knows that elements belonging to particular Namespaces are instructions, and any others are to be part of the result tree.

4.9 Resource Directory Description Language (RDDL)

As already stated, many people find Namespaces confusing because Namespace names are URIs, and so look just like web page addresses. Despite vociferous warnings not to, some people have built applications which rely on being able to resolve a Namespace URI into a real location, and certainly users have been known to click on Namespace URIs to see what will happen.

Given that all Namespaces have, or should have, some kind of defining material (or else what is the point of the Namespace?), be it a document designed for a human to read (e.g. a MIG), machine readable schemas, or combinations of the two; it appears to make sense for the Namespace URI to actually point (directly or indirectly) to that material.

If you try clicking on http://www.w3.org/1999/XSL/Transform (the XSLT namespace URI) you will see that the W3C have provided an HTML page that provides links to the XSLT recommendation, the XML recommendation, and the XML Namespaces recommendation. This certainly seems more polite and useful than simply sending back an error 404.

RDDL allows Namespace URI owners to point users to a document in a standard format. This acts as a directory with links to human and machine readable resources that can define and describe the elements belonging to a Namespace.

RDDL uses a simple mix of XHTML, for human navigable links, and XLink for machine navigable links. So the same RDDL document could point to schemas, implementation guidelines and even to executable code.

It has to be emphasised that the specification for RDDL has no official standing and has not been considered nor approved by any organization (e.g. W3C). It is possible that future versions of RDDL, or whatever may come to replace RDDL, will use RDF (Resource Description Framework) rather than XLink to define machine readable links. It also seems likely that RDDL will come to be viewed as a component of the Semantic Web and/or web services architectures.

Whatever, it is clear that XML Namespaces are coming to be considered by many to be more than just disambiguating markers. Whether this would have happened if Namespace values were not URIs is debatable.