XEP-0044: Full Namespace Support for XML Streams

Robert Norris
mailto:rob@cataclysm.cx
xmpp:rob@cataclysm.cx

2002-08-26
Version 0.1

Status	Type	Short Name
Deferred	Standards Track	N/A

A description of the use of namespaces within Jabber.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Jabber has traditionally supported a subset of the XML Namespaces specification [1]. The protocol has been restricted to using specific namespace prefixes.
This is convenient for client and server implementors, since they only need to check the element name to determine both the name and the context of the element. However, these restrictions mean that developers are unable to take advantage of some of the features that namespaces provide.
Many developers have expressed an interest in having Jabber fully support namespaces - a desire which is likely to increase as time goes on. This support consists of allowing any namespace prefix to be used with any namespace, and also to allow namespace prefixes to be pre-declared on the stream root.
This document outlines the semantics required for servers and clients to support namespaces fully, and also discusses implementation techniques and methods for providing compatibility with older "fixed-prefix" implementations.

2 Requirements and Protocol

A typical XML stream is a pair of XML documents, one for each direction of communication between the two peers. An simple example of these might look like this:

Listing 1: A typical XML stream

```
SEND: <stream:stream xmlns='jabber:client'
                     xmlns:stream='http://etherx.jabber.org/streams'
                     to='jabber.org'>
RECV: <stream:stream xmlns='jabber:client'
                     xmlns:stream='http://etherx.jabber.org/streams'
                     id='12345678'>
SEND: <iq type='get' to='jabber.org'>
        <query xmlns='jabber:iq:version'/>
      </iq>
RECV: <iq type='result' from='jabber.org'>
        <query xmlns='jabber:iq:version'>
          <name>jsm</name>
          <version>1.4.2</version>
          <os>Linux 2.4.19</os>
        </query>
      </iq>
```

Note that there may also be additional namespaces specified in the stream header, to select or inform of various server features:

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

Listing 2: A typical XML stream with stream options

```
SEND: <stream:stream xmlns='jabber:client'
                     xmlns:stream='http://etherx.jabber.org/streams'
                     xmlns:sasl='http://www.iana.org/assignments/sasl-
                         mechanisms'
                     to='jabber.org'>
RECV: <stream:stream xmlns='jabber:client'
                     xmlns:stream='http://etherx.jabber.org/streams'
                     xmlns:sasl='http://www.iana.org/assignments/sasl-
                         mechanisms'
                     id='12345678'>
      <sasl:mechanisms>
        <sasl:mechanism>PLAIN</sasl:mechanism>
        <sasl:mechanism>DIGEST-MD5</sasl:mechanism>
        <sasl:mechanism>EXTERNAL</sasl:mechanism>
      </sasl:mechanisms>
SEND: <iq type='get' to='jabber.org'>
        <query xmlns='jabber:iq:version'/>
      </iq>
RECV: <iq type='result' from='jabber.org'>
        <query xmlns='jabber:iq:version'>
          <name>jsm</name>
          <version>1.4.2</version>
          <os>Linux 2.4.19</os>
        </query>
      </iq>
```

Currently, the prefix for each namespace is fixed; it cannot vary at all, since implementations use it for matching. The desire is to be able to use arbitrary prefixes:

Listing 3: XML stream with arbitrary namespace prefixes (1)

```
SEND: <stream xmlns:app='jabber:client'
              xmlns='http://etherx.jabber.org/streams'
              to='jabber.org'>
RECV: <stream xmlns:app='jabber:client'
              xmlns='http://etherx.jabber.org/streams'
              id='12345678'>
SEND: <app:iq type='get' to='jabber.org'>
        <query xmlns='jabber:iq:version'/>
      </app:iq>
RECV: <app:iq type='result' from='jabber.org'>
        <query xmlns='jabber:iq:version'>
          <name>jsm</name>
          <version>1.4.2</version>
          <os>Linux 2.4.19</os>
        </query>
      </app:iq>
```

Also, since there exist streams in both directions, it should be possible for prefixes to differ between the two streams:

Listing 4: XML stream with arbitrary namespace prefixes

```
SEND: <stream xmlns:app='jabber:client'
              xmlns='http://etherx.jabber.org/streams'
              to='jabber.org'>
RECV: <stream:stream xmlns='jabber:client'
                     xmlns:stream='http://etherx.jabber.org/streams'
                     id='12345678'>
SEND: <app:iq type='get' to='jabber.org'>
        <query xmlns='jabber:iq:version'/>
      </app:iq>
RECV: <iq type='result' from='jabber.org'>
        <ver:query xmlns:ver='jabber:iq:version'>
          <ver:name>jsm</ver:name>
          <ver:version>1.4.2</ver:version>
          <ver:os>Linux 2.4.19</ver:os>
        </ver:query>
      </iq>
```

Additionally, it should be possible to declare namespaces on the stream header so that they don't need to be declared later:

Listing 5: XML stream with namespaces declared in the stream header

```
SEND: <stream:stream xmlns='jabber:client'
                     xmlns:stream='http://etherx.jabber.org/streams'
                     xmlns:ver='jabber:iq:version'
                     to='jabber.org'>
RECV: <stream:stream xmlns='jabber:client'
                     xmlns:stream='http://etherx.jabber.org/streams'
                     xmlns:ver='jabber:iq:version'
                     id='12345678'>
SEND: <iq type='get' to='jabber.org'>
        <ver:query/>
      </iq>
RECV: <iq type='result' from='jabber.org'>
        <ver:query>
          <ver:name>jsm</ver:name>
          <ver:version>1.4.2</ver:version>
          <ver:os>Linux 2.4.19</ver:os>
        </ver:query>
      </iq>
```

And of course, any combinations of these should be valid, as long as they conform to the XML Namespaces specification.

3 Implementation Notes

In order to implement namespaces correctly, implementations will need to check both the namespace of an element (or attribute), and its namespace, in order to match it. An implementation will need to maintain some sort of mapping between prefixes and namespaces, though some parsers, such as recent versions of Expat, can do this for the implementor.
Implementations should, wherever possible, adhere to the IETF maxim ”be liberal in what you accept, and conservative in what you send”. This means accepting any valid namespace prefix, but using only the traditional prefixes (i.e. ”stream” for ”http://etherx.jabber.org/streams”, ”sasl” for ”http://www.iana.org/assignments/sasl-mechanisms”, and no prefix for the application namespace). For servers, this has the added benefit of getting compatibility with non-namespace-aware clients for free.
In server components that may have to forward packets received from one stream to another stream, it may be necessary for the application namespace to be rewritten before the packet is forwarded. Examples of this are client-to-server and server-to-server components, which must convert ”jabber:client” and ”jabber:server” components, respectively, into ”jabber:component:accept” packets before they are forwarded to the router.