XEP-0189: Public Key Publishing

Peter Saint-Andre
mailto:stpeter@stpeter.im
xmpp:stpeter@jabber.org
https://stpeter.im/

Dirk Meyer
mailto:dmeyer@tzi.de
xmpp:dmeyer@jabber.org

Ian Paterson
mailto:ian.paterson@clientside.co.uk
xmpp:ian@zoofy.com

2010-07-15
Version 0.14

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines a method by which an entity can publish its public keys over XMPP.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

End-to-end encryption between XMPP clients and mutual authentication between a client and a server are desirable goals for the XMPP network. To enable these features, typically an XMPP user will need to possess a public key (or multiple keys), along with an associated private key that is not made public. The goal of this document is to provide simple methods for the exchange of public keys among XMPP users -- specifically, data formats that enable an XMPP entity to complete the following use cases:

- Publish its current public keys.
- Revoke keys that are no longer in use.
- Publish signed copies of its public keys (called attestations).
- Discover and retrieve public keys published by other entities.

2 Approach

This document assumes that each user will have a privately-generated key. A user **might** have more than one key (e.g., different keys for different clients) and the user's key **might** be signed by other entities, such as a certification authority (CA) or another user. However, the simplest case is a single key per user.
The data formats defined here are deliberately kept very simple, with the key itself represented as by the modulus and the public exponent, and any metadata represented using a few straightforward XML elements and attributes. This document does not use the 'http://www.w3.org/2000/09/xmldsig#' namespace as specified in XML Signature [1]; although that format is quite powerful, it introduces unnecessary levels of complexity for this use case. In addition, this document has no dependency on ASN.1, whose use is problematic on some computing platforms and languages (e.g., JavaScript).
We define three payload elements:

1. The <pubkey/> element (qualified by the 'urn:xmpp:pubkey:2' namespace) specifies a single public key for a user.
2. The <revoke/> element (qualified by the 'urn:xmpp:revoke:2' namespace) specifies old keys that a user once used but has now revoked.
3. The <attest/> element (qualified by the 'urn:xmpp:attest:2' namespace) specifies one or more signed copies of a user's public key.

An entity MAY support only public key publishing (the 'urn:xmpp:pubkey:2' namespace) and not support revocations and attestations.

[1] XML Signature Syntax and Processing <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>.

3 Public Keys

A single public key is contained in a <pubkey/> element qualified by the 'urn:xmpp:pubkey:2' namespace (see Namespace Versioning regarding the possibility of incrementing the version number), for which the defined child elements are as follows:

Element	Description
<begin/>	The UTC DateTime before which the key shall not be considered valid, formatted according to XMPP Date and Time Profiles (XEP-0082) XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>..
<end/>	The UTC DateTime after which the key shall not be considered valid, formatted according to XEP-0082.
<jid/>	The XMPP address associated with this key.
<rsakey/>	An element with three defined children for the modulus, public exponent, and fingerprint, where the input string to the hash function is the concatenation of the begin date, the end date, the JID, the string representation of the modulus, and the string representation of the public exponent, all encoded according to UTF-8 -- e.g., Base64(SHA-256(UTF8-encode(begin + end + jid + str(modulus) + str(exponent))).
<uri/>	A URI at which the key can be retrieved outside the XMPP network; this element is OPTIONAL.

The children of the <rsakey/> element are as follows.

Element	Description
<modulus/>	A large random number used as input to encryption and signing operations.
<publicExponent/>	The public exponent used as input to encryption and signing operations; RECOMMENDED to be "65537".
<print/>	A fingerprint for the key, where the algorithm used to generate the fingerprint is specified by the 'algo' attribute (whose default value is "sha-256").

An example follows.

Listing 1: A public key

```
<pubkey xmlns='urn:xmpp:pubkey:2'>
    <begin>2010-01-14T18:44:18Z</begin>
```

```
    <end>2011-01-14T18:44:18Z</end>
    <jid>alice@example.com</jid>
    <rsakey>
      <modulus>
1379023255526646700594339116162946782578210615394951216941818138582610132470

1538683909348374935399508656737300036572749687356120324044452513265157631177

7178489330603814456987700147603113573139002085123800563057708424072367754377

0906580036148700287401094143051242586181831057271078882862257109043483553172

66153
      </modulus>
      <publicExponent>65537</publicExponent>
      <print algo='sha-256'>eWGdcl+AzN0treQoRry+/
         zYqYJ7ZEAzwIvTossTURLw=</print>
    </rsakey>
  </pubkey>
```

4 Revocations

A user can revoke his own key. The revocation is contained in a <revoke/> element qualified by the 'urn:xmpp:revoke:2' namespace (see Namespace Versioning regarding the possibility of incrementing the version number), for which the defined child elements are as follows:

Element	Description
<key/>	The public key that was revoked.
<keyprint/>	The fingerprint of the public key that was revoked, where the algorithm is specified by the 'algo' attribute (which defaults to "sha-256").
<signature/>	A signature for the revocation, as described below.
<revocationprint/>	The fingerprint of the key that was used to sign the revocation, where the algorithm is specified by the 'algo' attribute (which defaults to "sha-256").
<revocationtime/>	The UTC DateTime at which the key was revoked, formatted according to XEP-0082.

The data that is signed MUST be the revoked public key (i.e., the XML character data of the <key/> element), the fingerprint of the revoked public key (i.e., the XML character data of the <keyprint/> element), the fingerprint of the key used to sign the revocation (i.e., the XML character data of the <revocationprint/> element), and the time of the revocation (i.e., the XML character data of the <revocationtime/> element), concatenated together with no spaces

or other additional characters.
As an example, consider a revocation of the key from Example 1, where the fingerprint of the revoked key is 13475c8e27399908b4447d7c52ab30822872832eba3a654f0d80e07fb4157673, the fingerprint of the key used to sign the revocation is becb78566783166f4a1a7c64e28dae288fe1a0f2825f6b593b336ce186c6b056, and the time of the revocation is 2009-12-14T20:49:16Z. The string to be signed would be as follows (where line breaks are not significant).


```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA25/Y7oonF3+pRjZCk0cQ
    lQFiUFvyMeUOnn01NLfqfBeT4xjahsxaYFPd+V8jIHtpO3nbkpfrBh5aLPCkqDzS
    UFvmLaVSIMUt085kcT/Yv72BUWUGaISt7swSNfictRbVZ0k8TbXUDLIAYWN8lsie
    NVxL13PS9pu3WNoORuKAV5RuzDS3djnjb6fti7fTwEUpQVwRrJXY6jNhv4c4zE3x
    Pjm48gIlHbZrxuWNrXnKT6uwXnr6PJ6O3YspferJG1oA3kmOwm41yWDkalbKgkNZ
    WeS+ahB/i9LIG+41CdekPpI0Kr0Ll9X7+zaPV6xzjlG8dYy7ZGm4eTL9STuWlPdL
    +wIDAQAB13475c8e27399908b4447d7c52ab30822872832eba3a654f0d80e07f
    b4157673becb78566783166f4a1a7c64e28dae288fe1a0f2825f6b593b336ce1
    86c6b0562009-12-14T20:49:16Z
```


An example follows.

Listing 2: A revocation


```
<revocation xmlns='urn:xmpp:revoke:2'>
  <key>
    MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA25/Y7oonF3+pRjZCk0cQ
    lQFiUFvyMeUOnn01NLfqfBeT4xjahsxaYFPd+V8jIHtpO3nbkpfrBh5aLPCkqDzS
    UFvmLaVSIMUt085kcT/Yv72BUWUGaISt7swSNfictRbVZ0k8TbXUDLIAYWN8lsie
    NVxL13PS9pu3WNoORuKAV5RuzDS3djnjb6fti7fTwEUpQVwRrJXY6jNhv4c4zE3x
    Pjm48gIlHbZrxuWNrXnKT6uwXnr6PJ6O3YspferJG1oA3kmOwm41yWDkalbKgkNZ
    WeS+ahB/i9LIG+41CdekPpI0Kr0Ll9X7+zaPV6xzjlG8dYy7ZGm4eTL9STuWlPdL
    +wIDAQAB
  </key>
  <keyprint>iof8LEqhKyItN0TsVgWuIFBhEwHsORgp6nocM1WAUmk=</keyprint>
  <revocationprint>
      becb78566783166f4a1a7c64e28dae288fe1a0f2825f6b593b336ce186c6b056
      </revocationprint>
  <revocationtime>2009-12-14T20:49:16Z</revocationtime>
  <signature>
    iEYEARECAAYFAksnvfsACgkQNL8k5A2w/vyVHgCglI5fPQfcu1e5PTA3bMPKiL0F
    J5cAnirpKzJ45OD+03b66UHoIosKQ31O
    =PGFT
  </signature>
</revocation>
```

5 Attestations

Whether in the context of the public key infrastructure (PKI) or a web of trust, a user might want to publish signed copies of his public key, where the signer might be another user or a trusted third party such as a certification authority (CA). The signing key could even be another key owned by the user (e.g., a primary key used to sign a secondary key associated with a particular device controlled by the user). Furthermore, the signing material might be an RSA key, a DSA key, an X.509 certificate, an OpenPGP key, or any other format.
Each signed copy is contained in an <attest/> element qualified by the 'urn:xmpp:attest:2' namespace (see Namespace Versioning regarding the possibility of incrementing the version number), for which the defined child elements are as follows:

Element	Description
<keyprint/>	The fingerprint of the key that was signed, where the algorithm is specified by the 'algo' attribute (which defaults to "sha-256").
<signature/>	The signature itself, as described below.
<signerjid/>	The XMPP address of the signer.
<signerprint/>	The fingerprint of the signer's key, where the algorithm is specified by the 'algo' attribute (which defaults to "sha-256").
<signtime/>	The UTC DateTime at which the key was signed, formatted according to XEP-0082.

The data that is signed MUST be the user's public key (i.e., the XML character data of the <key/> element), the fingerprint of the user's public key (i.e., the XML character data of the <keyprint/> element), the signer's JID (i.e., the XML character data of the <signerjid/> element), the fingerprint of the signer's key (i.e., the XML character data of the <signerprint/> element), and the time of the attestation (i.e., the XML character data of the <signtime/> element), concatenated together with no spaces or other additional characters.
As an example, consider an attestation of the key from Example 1, where the signer's JID is stpeter@jabber.org, the fingerprint of the signer's key is "becb78566783166f4a1a7c64e28dae288fe1a0f2825f6b593b336ce186c6b056", and the time of the attestation is 2009-12-14T18:43:00Z. The string to be signed would be as follows (where line breaks are not significant).


```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA25/Y7oonF3+pRjZCk0cQ
    lQFiUFvyMeUOnn01NLfqfBeT4xjahsxaYFPd+V8jIHtpO3nbkpfrBh5aLPCkqDzS
    UFvmLaVSIMUt085kcT/Yv72BUWUGaISt7swSNfictRbVZ0k8TbXUDLIAYWN8lsie
    NVxL13PS9pu3WNoORuKAV5RuzDS3djnjb6fti7fTwEUpQVwRrJXY6jNhv4c4zE3x
    Pjm48gIlHbZrxuWNrXnKT6uwXnr6PJ6O3YspferJG1oA3kmOwm41yWDkalbKgkNZ
    WeS+ahB/i9LIG+41CdekPpI0Kr0Ll9X7+zaPV6xzjlG8dYy7ZGm4eTL9STuWlPdL
    +wIDAQAB13475c8e27399908b4447d7c52ab30822872832eba3a654f0d80e07f
    b4157673stpeter@jabber.orgbecb78566783166f4a1a7c64e28dae288fe1a0
```

```
    f2825f6b593b336ce186c6b0562009-12-14T18:43:00Z
```

Listing 3: RSA public key signed by OpenPGP key

```
<attest xmlns='urn:xmpp:attest:2'>
  <keyprint>iof8LEqhKyItN0TsVgWuIFBhEwHsORgp6nocM1WAUmk=</keyprint>
  <signature>
    iEYEARECAAYFAksnyJ0ACgkQNL8k5A2w/vy9DgCfc1eOYZlp1P456NkjgvCtJaRX
    Wi4AoIGhfzr7XMzeOzOBi9A/bxopH8O4
    =MMuz
  </signature>
  <signerjid>stpeter@jabber.org</signerjid>
  <signerprint>
     becb78566783166f4a1a7c64e28dae288fe1a0f2825f6b593b336ce186c6b056
     </signerprint>
  <signtime>2009-12-14T18:43:00Z</signtime>
</attest>
```

6 Publication via PEP

A user SHOULD publish public keys, attestations, and revocations via Personal Eventing Protocol (XEP-0163) [2], in particular adhering to the best practices defined in Best Practices for Persistent Storage of Public Data via Publish-Subscribe (XEP-0222) [3].
Any other authorized entity can then receive notifications related to the user's public keys, and retrieve keys, attestations, and revocations.
If the user has only one public key (the simplest and most common case), then the pubkey PEP node SHOULD have only one item, with an ItemID of "current".

7 Requesting a Public Key Directly

If it is not possible for the user to publish his key via PEP, or for a contact to retrieve the user's key via PEP, the contact MAY request the user's key directly by sending an <iq/> of type 'get' to the user's particular full JID, containing an empty <pubkey/> element qualified by the 'urn:xmpp:pubkey:2' namespace (see Namespace Versioning regarding the possibility of incrementing the version number).

Listing 4: Public key request

```
<iq from='maineboy@jabber.org/bar'
    to='peter@jabber.org/foo'
```

[2] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[3] XEP-0222: Best Practices for Persistent Storage of Public Data via Publish-Subscribe <https://xmpp.org/extensions/xep-0222.html>.

```
    id='hfgt654s'
    type='get'>
  <pubkey xmlns='urn:xmpp:pubkey:2'/>
</iq>
```

The receiving client then return its public key (which might be different from the overall key for the user, e.g. a key for only a particular device).

Listing 5: Public key response

```
<iq from='peter@jabber.org/foo'
    to='maineboy@jabber.org/bar'
    id='hfgt654s'
    type='result'>
  <pubkey xmlns='urn:xmpp:pubkey:2'>
    <begin>2009-12-11T20:12:37</begin>
    <end>2010-12-11T23:59:59</end>
    <jid>peter@jabber.org</jid>
    <key>
      MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA25/Y7oonF3+pRjZCk0cQ
      lQFiUFvyMeUOnn01NLfqfBeT4xjahsxaYFPd+V8jIHtpO3nbkpfrBh5aLPCkqDzS
      UFvmLaVSIMUt085kcT/Yv72BUWUGaISt7swSNfictRbVZ0k8TbXUDLIAYWN8lsie
      NVxL13PS9pu3WNoORuKAV5RuzDS3djnjb6fti7fTwEUpQVwRrJXY6jNhv4c4zE3x
      Pjm48gIlHbZrxuWNrXnKT6uwXnr6PJ6O3YspferJG1oA3kmOwm41yWDkalbKgkNZ
      WeS+ahB/i9LIG+41CdekPpI0Kr0Ll9X7+zaPV6xzjlG8dYy7ZGm4eTL9STuWlPdL
      +wIDAQAB
    </key>
    <print>13475
       c8e27399908b4447d7c52ab30822872832eba3a654f0d80e07fb4157673</
       print>
  </pubkey>
</iq>
```

8 Sending a Public Key Directly

An entity might need to send its public key to another entity, for example if it has generated a new key but does not have a way to publish the new key (or does not wish to publish the key in a world-readable fashion). In this case the entity MAY include the key directly in a <message/> stanza.

Listing 6: Sending a key in a message

```
<message from='peter@jabber.org/foo'
         to='maineboy@jabber.org/bar'>
  <pubkey xmlns='urn:xmpp:pubkey:2'>
    <begin>2009-12-11T20:12:37</begin>
    <end>2010-12-11T23:59:59</end>
```

```
    <jid>peter@jabber.org</jid>
    <key>
      MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA25/Y7oonF3+pRjZCk0cQ
      lQFiUFvyMeUOnn01NLfqfBeT4xjahsxaYFPd+V8jIHtpO3nbkpfrBh5aLPCkqDzS
      UFvmLaVSIMUt085kcT/Yv72BUWUGaISt7swSNfictRbVZ0k8TbXUDLIAYWN8lsie
      NVxL13PS9pu3WNoORuKAV5RuzDS3djnjb6fti7fTwEUpQVwRrJXY6jNhv4c4zE3x
      Pjm48gIlHbZrxuWNrXnKT6uwXnr6PJ6O3YspferJG1oA3kmOwm41yWDkalbKgkNZ
      WeS+ahB/i9LIG+41CdekPpI0Kr0Ll9X7+zaPV6xzjlG8dYy7ZGm4eTL9STuWlPdL
      +wIDAQAB
    </key>
    <print>13475
        c8e27399908b4447d7c52ab30822872832eba3a654f0d80e07fb4157673</
        print>
  </pubkey>
</message>
```

9 Signalling Key Generation

There are several situations in which it is helpful for an entity to signal that it is currently generating a key. For example, a client that does not have access to permanent storage might generate a key on startup, but key generation might not be complete when the client sends initial presence upon establishing an XMPP session. In this case the client might signal support for the public key format in the entity capabilities data that it includes in its initial presence broadcast, but also include an indication that it is currently generating a key.

Listing 7: Key generation in progress

```
<presence from='peter@jabber.org/foo'>
  <generating xmlns='urn:xmpp:pubkey:2'/>
</presence>
```

After key generation is complete, the entity could publish the new key to the appropriate PEP node (if available) and send updated presence without the <generating/> extension.

Listing 8: Key generation completed

```
<presence from='peter@jabber.org/foo'/>
```

10 Determining Support

To advertise its support for public keys, revocations, and attestations, when replying to Service Discovery (XEP-0030) [4] information requests an entity MUST return features for 'urn:xmpp:pubkey:2', 'urn:xmpp:revoke:2', and 'urn:xmpp:attest:2' respectively.

[4] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

In order for an application to determine whether an entity supports this protocol, where possible it SHOULD use the dynamic, presence-based profile of service discovery defined in Entity Capabilities (XEP-0115) [5]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

11 Security Considerations

The reliable association between a user or entity and its public keys is beyond the scope of this document. However, each client SHOULD maintain its own secure library of the public keys it associates with other users.

12 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [6].

13 XMPP Registrar Considerations

13.1 Protocol Namespaces

This specification defines the following XML namespaces:

- urn:xmpp:attest:2
- urn:xmpp:pubkey:2
- urn:xmpp:revoke:2

Upon advancement of this specification to a status of Draft, the XMPP Registrar [7] shall add these namespaces to the registry located at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [8].

[5] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

[6] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[7] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.

[8] XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.

13.2 Protocol Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

14 XML Schemas

14.1 attest

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:attest:2'
    xmlns='urn:xmpp:attest:2'
    elementFormDefault='qualified'>

  <xs:element name='attest'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='keyprint' type='fingerprintType' minOccurs='
            1' maxOccurs='1'/>
        <xs:element name='signature' type='xs:string' minOccurs='1'
            maxOccurs='1'/>
        <xs:element name='signerjid' type='xs:string' minOccurs='1'
            maxOccurs='1'/>
        <xs:element name='signerprint' type='fingerprintType'
            minOccurs='1' maxOccurs='1'/>
        <xs:element name='signtime' type='xs:dateTime' minOccurs='1'
            maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='fingerprintType'>
    <xs:simpleContent>
      <xs:extension base='xs:string'>
        <xs:attribute name='algo' type='xs:NMTOKEN' use='optional'
            default='sha-256'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

</xs:schema>
```

14.2 pubkey

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:pubkey:2'
    xmlns='urn:xmpp:pubkey:2'
    elementFormDefault='qualified'>

  <xs:element name='pubkey'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='begin' type='xs:dateTime' minOccurs='1'
            maxOccurs='1'/>
        <xs:element name='end' type='xs:dateTime' minOccurs='1'
            maxOccurs='1'/>
        <xs:element name='jid' type='xs:string' minOccurs='1'
            maxOccurs='1'/>
        <xs:element name='rsakey' type='rsaKeyType' minOccurs='1'
            maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='rsaKeyType'>
    <xs:sequence>
      <xs:element name='modulus' type='xs:string' minOccurs='1'
          maxOccurs='1'/>
      <xs:element name='publicExponent' type='xs:string' minOccurs='1'
           maxOccurs='1'/>
      <xs:element name='print' type='fingerprintType' minOccurs='1'
          maxOccurs='1'/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='fingerprintType'>
    <xs:simpleContent>
      <xs:extension base='xs:string'>
        <xs:attribute name='algo' type='xs:NMTOKEN' use='optional'
            default='sha-256'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

</xs:schema>
```

14.3 revoke

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:pubkey:2'
    xmlns='urn:xmpp:pubkey:2'
    elementFormDefault='qualified'>

  <xs:element name='revocation'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='key' type='xs:string' minOccurs='1'
            maxOccurs='1'/>
        <xs:element name='keyprint' type='fingerprintType' minOccurs='
            1' maxOccurs='1'/>
        <xs:element name='signature' type='xs:string' minOccurs='1'
            maxOccurs='1'/>
        <xs:element name='revocationprint' type='fingerprintType'
            minOccurs='1' maxOccurs='1'/>
        <xs:element name='revocationtime' type='xs:dateTime' minOccurs
            ='1' maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='fingerprintType'>
    <xs:simpleContent>
      <xs:extension base='xs:string'>
        <xs:attribute name='algo' type='xs:NMTOKEN' use='optional'
            default='sha-256'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

</xs:schema>
```