# XEP-0227: Portable Import/Export Format for XMPP-IM Servers


Magnus Henoch
mailto:henoch@dtek.chalmers.se
xmpp:legoscia@jabber.cd.chalmers.se

Waqas Hussain
mailto:waqas20@gmail.com
xmpp:waqas@jaim.at

Matthew Wild
mailto:mwild1@gmail.com
xmpp:me@matthewwild.co.uk



2021-06-02
Version 1.1

| Status | Type | Short Name |
| --- | --- | --- |
| Draft | Standards Track | pie |



This document specifies a file format for importing and exporting user data to and from XMPP-IM servers.

# Legal

## Copyright



## Permissions



## Warranty



## Liability



## Conformance

# Contents

# 1 Introduction

Different implementations of XMPP-IM servers store user data in various ways, and many implementations have more than one storage format. This leads to problems when a server administrator wants to switch to another implementation or storage format -- the implementation is as likely as not to have an import mechanism that can read the user data in its current form. This document attempts to solve that problem by defining a common file format for import and export of user data in XMPP-IM servers.

# 2 Requirements

The following constraints are imposed on this standard:

- The file format is XML-based.
  XMPP-IM servers already have tools to process XML data. This also allows extension of the format using namespaces. Furthermore, some of the data that needs to be stored is by definition already in XML form.

- The data layout is flexible.
  The data is contained in a single XML document; however, it can be split into several files using XInclude [1].

- All user data is stored, but no server configuration data.
  User data has similar form throughout the XMPP world, but server configuration is implementation-specific. Therefore this specification does not attempt to transfer any aspects of the server configuration from one server to another.
  Furthermore, the contents of MUC, Pubsub and other services are out of scope for this specification.

- Multiple virtual hosts are supported.
  Many server implementations can serve several hostnames in a single server instance. Thus this specification allows storing data from several virtual hosts.

# 3 Glossary

**Exporting server** The XMPP-IM server writing its user data to files, following this specification.

[1] XML Inclusions (XInclude) 1.0 <http://www.w3.org/TR/xinclude/>.

**Importing server** The XMPP-IM server reading data from such files.

# 4 File format

Data is contained in an XML document, whose root element is <server-data/> qualified by the 'urn:xmpp:pie:0' namespace (see Protocol Namespaces regarding issuance of one or more permanent namespaces).

Listing 1: The root element

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  [ ... ]
</server-data>
```

At any point in the file, an exporting server may put elements qualified by a namespace not mentioned in this specification. The exported data SHOULD be meaningful without the extensions. An importing server that encounters a namespace that it doesn't understand, or otherwise is unable to import all given data, SHOULD ignore the unknown data, SHOULD notify the operator, and MAY offer to terminate the process.
At any point in the file, an exporting server may put an XInclude <include/> element; see Use of XInclude.

## 4.1 Hosts

The child elements of the <server-data/> elements are <host/> elements. Each <host/> element describes a virtual host, and has a 'jid' attribute that contains its JID.
An importing server MAY automatically adjust its list of virtual hosts to fit the ones present in the data being imported. If it does not, it SHOULD notify the operator about any mismatch.

Listing 2: The host element

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    [ ... ]
  </host>
  <host jid='montague.net'>
    [ ... ]
  </host>
</server-data>
```

### 4.2 Users

Each user is represented by a <user/> element under the <host/> element. The <user/> element MUST have a 'name' attribute, which contains the node part of the user's JID.
If the plaintext password of the user is known, it MAY be included in the 'password' attribute, although this is not recommended from a security perspective. For more information see Security Considerations. See also the SCRAM credentials section for an alternative.

Listing 3: The user element

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      [ ... ]
    </user>
  </host>
  <host jid='montague.net'>
    <user name='romeo'>
      [ ... ]
    </user>
  </host>
</server-data>
```

### 4.3 SCRAM credentials

Authentication secrets may be included that allow for authentication using the SCRAM family of mechanisms, as defined in RFC 5802 [2].
Each set of credentials should be encapsulated within a <scram-credentials/> element in the 'urn:xmpp:pie:0#scram' namespace, and contained within the relevant <user/> element. The element should have a 'mechanism' attribute specifying the registered name of the mechanism that the credentials are used for (always without the "-PLUS" suffix), e.g. 'SCRAM-SHA-1'. The element MUST contain a single occurrence of each of the following child elements:

- <iter-count/>: containing the SCRAM iteration count, e.g. '10000'. This must be a positive integer without leading zeros.
- <salt/>: containing the base64-encoded salt.
- <server-key/>: containing the base64-encoded ServerKey defined by SCRAM.
- <stored-key/>: containing the base64-encoded StoredKey defined by SCRAM.

[2] RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms <http://tools.ietf.org/html/rfc5802>.

There may be multiple occurrences of <scram-credentials/> for a single user, however they MUST all have a unique 'mechanism' attribute.

Listing 4: Including a user's SCRAM credentials

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      <scram-credentials xmlns='urn:xmpp:pie:0#scram' mechanism='SCRAM
          -SHA-1'>
        <iter-count>100000</iter-count>
        <salt>
            TmFDbE5hQ2xOYUNsTmFDbE5hQ2xOYUNsTmFDbE5hQ2xOYUNsTmFDbE5hQ2wK
            </salt>
        <server-key>0pXWGK0GZJ6TR73AIUN3ITYtA1g=</server-key>
        <stored-key>Q6qT/SbybblGCZz8e8eSfCJOQic=</stored-key>
      </scram-credentials>
    </user>
  </host>
</server-data>
```

Be aware of the Security Considerations when including credentials in a data export. Even though SCRAM credentials are stored in a hashed form, leaking them still allows an attacker to impersonate the user to other servers employing the same SCRAM parameters, and it also allows for offline dictionary or brute-force attacks.

### 4.4 Rosters

Each <user/> element SHOULD contain the user's roster in the form of a <query/> element qualified by the 'jabber:iq:roster' namespace. This element contains the user's roster in the same format as when retrieving the roster from the server, as described in section 7.3 of XMPP IM [3].

Listing 5: The roster

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      <query xmlns='jabber:iq:roster'>
        <item jid='romeo@montague.net'
              name='Romeo'
              subscription='both'>
```

[3] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.

```
          <group>Friends</group>
        </item>
      </query>
    </user>
  </host>
</server-data>
```

### 4.5 Offline Messages

If the exporting server stores messages received while the user was offline, it SHOULD include an <offline-messages/> element as a child of the <user/> element. This element contains all the stored messages to the user, if any, as <message/> elements qualified by the 'jabber:client' namespace, starting with the oldest.

Listing 6: Offline messages

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      <offline-messages>
        <message xmlns='jabber:client'
                 from='romeo@montague.net/orchard'
                 to='juliet@capulet.com/balcony'
                 type='chat'>
          <body>Neither, fair saint, if either thee dislike.</body>
          <delay xmlns='urn:xmpp:delay'
                 from='capulet.com'
                 stamp='1469-07-21T00:32:29Z'>
            Offline Storage
          </delay>
        </message>
      </offline-messages>
    </user>
  </host>
</server-data>
```

### 4.6 Private XML Storage

Private data stored by the server as specified in Private XML Storage (XEP-0049) [4] is represented in this format by including a <query/> element qualified by the 'jabber:iq:private' namespace as a child of the <user/> element. This <query/> element in turn contains all elements saved in private XML storage.

[4] XEP-0049: Private XML Storage <https://xmpp.org/extensions/xep-0049.html>.

Listing 7: Private XML Storage

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='shakespeare.lit'>
    <user name='hamlet'>
      <query xmlns="jabber:iq:private">
        <exodus xmlns="exodus:prefs">
          <defaultnick>Hamlet</defaultnick>
        </exodus>
      </query>
    </user>
  </host>
</server-data>
```

## 4.7 vCards

By vcard-temp (XEP-0054) [5], users can store vCards on the server. In this specification, vCards are child elements of the <user/> element, namely a <vCard/> element qualified by the 'vcard-temp' namespace.

Listing 8: vCards

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      <vCard xmlns='vcard-temp'>
        <FN>Juliet Capulet</FN>
      </vCard>
    </user>
  </host>
</server-data>
```

## 4.8 Privacy Lists

Privacy lists, as specified in Privacy Lists (XEP-0016) [6], are represented in this format by including a <query/> element qualified by the 'jabber:iq:privacy' namespace as a child of the <user/> element. This element should contain all privacy lists associated with the user. A default privacy list, if set, is specified by including a <default/> element as a child of the <query/> element.

Listing 9: Privacy lists

[5] XEP-0054: vcard-temp <https://xmpp.org/extensions/xep-0054.html>.
[6] XEP-0016: Privacy Lists <https://xmpp.org/extensions/xep-0016.html>.

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      <query xmlns='jabber:iq:privacy'>
        <default name='public'/>
        <list name='public'>
          <item type='jid'
                value='tybalt@example.com'
                action='deny'
                order='1'/>
          <item action='allow' order='2'/>
        </list>
        <list name='private'>
          <item type='subscription'
                value='both'
                action='allow'
                order='10'/>
          <item action='deny' order='15'/>
        </list>
      </query>
    </user>
  </host>
</server-data>
```

### 4.9 Incoming Subscription Requests

Each <user/> element SHOULD contain pending incoming subscription requests associated with the user's account. Incoming subscription requests are represented by including <presence/> elements qualified by the 'jabber:client' namespace with the 'type' attribute set to a value of 'subscribe' as children of the <user/> element.

Listing 10: Incoming subscription requests

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      <presence xmlns='jabber:client'
                type='subscribe'
                id='xk3h1v69'
                from='romeo@montague.net'>
        <nick xmlns="http://jabber.org/protocol/nick">Romeo</nick>
      </presence>
      <presence xmlns='jabber:client'
                type='subscribe'
                from='mercutio@montague.net'/>
    </user>
```

```
  </host>
</server-data>
```

## 4.10 Personal Eventing Protocol

A user's PEP data (as defined in Personal Eventing Protocol (XEP-0163) [7]) SHOULD be included if known.
Node configuration and the actual node data are encapsulated separately, as described below. A typical export that contains both node configuration and the actual data contained within the node, will include two <pubsub/> elements (qualified by different namespaces).
Many server implementations include support for additional pubsub features from Publish-Subscribe (XEP-0060) [8] beyond those required by XEP-0163. This specification aims to preserve this additional data also, when it is present and supported by both servers.

### 4.10.1 PEP node configuration

Within the <user/> element there should be a single <pubsub/> element qualified by the 'http://jabber.org/protocol/pubsub#owner' namespace (note the '#owner' suffix). Within this element, there MUST be one <configure/> element for each exported node, with the node's name in the 'node' attribute. There MAY be additional elements included, at most one per node of each kind: <subscriptions/> and <affiliations>, following the syntax defined in XEP-0060.
The format of the <configure/> is a Data Forms (XEP-0004) [9] data form, typically containing the fields documented in XEP-0060, encoding the configuration of the named node.
As a general rule, importers SHOULD ignore node configuration options that the target server implementation doesn't recognise, to allow porting data between different implementations even in the presence of custom extensions. Exceptions to this requirement may be made for imports that are expected to be lossless, for example if the user has specifically requested a lossless import, or if the importer recognises certain configuration fields as critical to protect the node's security or integrity.

### 4.10.2 PEP node items

Within the <user/> element there should be a single <pubsub/> element qualified by the 'http://jabber.org/protocol/pubsub' namespace (note the lack of any suffix). Within this element, there MUST be one <items/> element for each exported node, with the node's name in the 'node' attribute.
Any node listed in this element MUST have a corresponding configuration included as described in the previous section.

[7] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[8] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[9] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.

Each <items> element MUST contain zero or more <item/> elements as defined by XEP-0060. This example demonstrates an export for a user who has two nodes: a private bookmarks node with two bookmarks, and a public nickname node containing a single item.

Listing 11: Romeo's exported PEP data

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='romeo'>
      <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
        <configure node='urn:xmpp:bookmarks:1'>
          <x xmlns='jabber:x:data' type='form'>
            <field var='FORM_TYPE' type='hidden'>
              <value>http://jabber.org/protocol/pubsub#node_config</
                  value>
            </field>
            <field var='pubsub#access_model'>
              <value>whitelist</value>
            </field>
          </x>
        </configure>
        <affiliations node='urn:xmpp:bookmarks:1'>
          <affiliation jid='mercutio@example.net' affiliation='member'
              />
        </affiliations>
        <subscriptions node='urn:xmpp:bookmarks:1'>
          <subscription jid='mercutio@example.net' subscription='
              subscribed' subid='123-abc'/>
        </subscriptions>
        <configure node='http://jabber.org/protocol/nick'>
          <x xmlns='jabber:x:data' type='form'>
            <field var='FORM_TYPE' type='hidden'>
              <value>http://jabber.org/protocol/pubsub#node_config</
                  value>
            </field>
            <field var='pubsub#access_model'>
              <value>open</value>
            </field>
          </x>
        </configure>
      </pubsub>
      <pubsub xmlns='http://jabber.org/protocol/pubsub'>
        <items node='urn:xmpp:bookmarks:1'>
          <item id='theplay@conference.shakespeare.lit'>
            <conference xmlns='urn:xmpp:bookmarks:1'
                        name='The␣Play&apos;s␣the␣Thing'
                        autojoin='true'>
              <nick>Romeo</nick>
```

```
          </conference>
        </item>
        <item id='orchard@conference.shakespeare.lit'>
          <conference xmlns='urn:xmpp:bookmarks:1'
                      name='The␣Orchard'
                      autojoin='1'>
            <nick>Romeo</nick>
          </conference>
        </item>
      </items>
      <items node='http://jabber.org/protocol/nick'>
        <item id='current'>
          <nick xmlns='http://jabber.org/protocol/nick'>Romy</nick>
        </item>
      </items>
    </pubsub>
  </user>
 </host>
</server-data>
```

### 4.11 Message Archive

A user's Message Archive Management (XEP-0313) [10] message archive MAY be included in an export. If included, they MUST be formatted as a series of XEP-0313 <result/> elements within an <archive/> element qualified by the 'urn:xmpp:pie:0#mam' namespace. The result elements MUST be in chronological order (from oldest to newest).

Listing 12: Juliet's exported message archive

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet'>
      <archive xmlns='urn:xmpp:pie:0#mam'>
        <result xmlns='urn:xmpp:mam:2' id='28482-98726-73623'>
            <forwarded xmlns='urn:xmpp:forward:0'>
              <delay xmlns='urn:xmpp:delay' stamp='2010-07-10
                  T23:08:25Z'/>
              <message xmlns='jabber:client'
                to='juliet@capulet.lit/balcony'
                from='romeo@montague.lit/orchard'
                type='chat'>
                <body>Call me but love, and I'll␣be␣new␣baptized;␣
                   Henceforth␣I␣never␣will␣be␣Romeo.</body>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣</message>
␣␣␣␣␣␣␣␣␣␣␣␣</forwarded>
```

[10] XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.

```
␣␣␣␣␣␣␣␣␣␣</result>
␣␣␣␣␣␣␣␣␣␣<result␣xmlns='urn:xmpp:mam:2'␣id='5d398-28273-f7382'>
␣␣␣␣␣␣␣␣␣␣␣␣<forwarded␣xmlns='urn:xmpp:forward:0'>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣<delay␣xmlns='urn:xmpp:delay'␣stamp='2010-07-10
    T23:09:32Z'/>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣<message␣xmlns='jabber:client'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣to='romeo@montague.lit/orchard'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣from='juliet@capulet.lit/balcony'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣type='chat'␣id='8a54s'>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<body>What␣man␣art␣thou␣that␣thus␣bescreen'd in night
    so stumblest on my counsel?</body>
              </message>
            </forwarded>
          </result>
      </archive>
    </user>
  </host>
</server-data>
```

# 5 Use of XInclude

An exporting server may split the data in several files by using the XInclude <include/> element. An importing server MUST support <include/> elements having an 'href' attribute containing a relative URI, having no 'parse' attribute, and having no 'xpointer' attribute; it MAY support other kinds of <include/> elements. An exporting server SHOULD NOT include and an importing server SHOULD NOT process <include/> elements which are descendants, but not children of the <user/> element (since these may be part of user data).

## 5.1 File and Directory Layout

If an exporting server chooses to split the data into several files, it SHOULD use the following scheme:
The main file contains the <server-data/> element, which contains nothing but one <include/> element for each host. The file included for a certain host is placed in the same directory as the main file, and is named by appending ".xml" to the JID of the host, e.g. "capulet.com.xml".

Listing 13: The main file, which includes host files

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'
             xmlns:xi='http://www.w3.org/2001/XInclude'>
  <xi:include href='capulet.com.xml'/>
  <xi:include href='montague.net.xml'/>
</server-data>
```

Each host file contains a <host/> element, which contains nothing but one <include/> element for each user of the host. The file included for a certain user is placed in a subdirectory whose name is the JID of the host, and is named by appending ".xml" to the node part of the user's JID, e.g. "capulet.com/juliet.xml".

Listing 14: The host file, which includes user files

```
<?xml version='1.0' encoding='UTF-8'?>
<host xmlns='urn:xmpp:pie:0'
      xmlns:xi='http://www.w3.org/2001/XInclude'
      jid='capulet.com'>
  <xi:include href='capulet.com/juliet.xml'/>
  <xi:include href='capulet.com/mercutio.xml'/>
</host>
```

Each user file contains a <user/> element, and includes all data relating to that user.

Listing 15: The user file

```
<?xml version='1.0' encoding='UTF-8'?>
<user xmlns='urn:xmpp:pie:0'
      name='juliet'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@montague.net'
          name='Romeo'
          subscription='both'>
      <group>Friends</group>
    </item>
  </query>
  <vCard xmlns='vcard-temp'>
    <FN>Juliet Capulet</FN>
  </vCard>
  <offline-messages>
    <message xmlns='jabber:client'
             from='romeo@montague.net/orchard'
             to='juliet@capulet.com/balcony'
             type='chat'>
      <body>Neither, fair saint, if either thee dislike.</body>
      <delay xmlns='urn:xmpp:delay'
             from='capulet.com'
             stamp='1469-07-21T00:32:29Z'>
        Offline Storage
      </delay>
    </message>
  </offline-messages>
</user>
```

The definition of JIDs ensures that this generates valid file names on traditional Unix-like file systems, except for possible length constraints. However, various constraints may force an

exporting server to alter this scheme. In any case, the importing server MUST NOT rely on this layout, but MUST do proper XInclude processing.

# 6 Security Considerations

Exported data files are to be handled with care, since they contain data that users expect to be protected, in particular passwords. An exporting server SHOULD make sure that the generated file is not accessible to unauthorized persons, e.g. by enforcing strict file permissions. It may also apply suitable encryption before storing or transmitting the data.
XInclude <include/> elements which are indirect descendants of the <user/> element SHOULD be treated as opaque user data, and SHOULD NOT be processed.

# 7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [11].

# 8 XMPP Registrar Considerations

## 8.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:pie:0

The XMPP Registrar [12] includes the foregoing namespace in its registry at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [13].

## 8.2 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

[11] 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/>.

[12] 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/>.

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

number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

## 9 XML Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:pie:0'
    xmlns='urn:xmpp:pie:0'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0227: http://xmpp.org/extensions/xep-0227.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='server-data'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='host' maxOccurs='unbounded'/>
        <xs:any namespace='##other' minOccurs='0' maxOccurs='unbounded
            '/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='host'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='user' maxOccurs='unbounded'/>
        <xs:any namespace='##other' minOccurs='0' maxOccurs='unbounded
            '/>
      </xs:sequence>
      <xs:attribute name='jid' type='xs:string' use='required'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='user'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='offline-messages' minOccurs='0' maxOccurs='1'
            />
```

```
        <xs:any namespace='##other' minOccurs='0' maxOccurs='unbounded
            '/>
      </xs:sequence>
      <xs:attribute name='name' type='xs:string' use='required'/>
      <xs:attribute name='password' type='xs:string' use='optional'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='offline-messages'>
    <xs:complexType>
      <xs:sequence>
        <xs:any namespace='##other' minOccurs='0' maxOccurs='unbounded
            '/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```