



# XMPP

## 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>

2010-03-12  
Version 1.0

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

This XMPP Extension Protocol is copyright © 1999 – 2017 by the [XMPP Standards Foundation](#) (XSF).

## Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

## Warranty

## NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. ##

## Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

## Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF's Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy>) or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Requirements</b>	<b>1</b>
<b>3</b>	<b>Glossary</b>	<b>1</b>
<b>4</b>	<b>File format</b>	<b>2</b>
4.1	Hosts . . . . .	2
4.2	Users . . . . .	3
4.3	Rosters . . . . .	3
4.4	Offline Messages . . . . .	4
4.5	Private XML Storage . . . . .	4
4.6	vCards . . . . .	5
4.7	Privacy Lists . . . . .	5
4.8	Incoming Subscription Requests . . . . .	6
<b>5</b>	<b>Use of XInclude</b>	<b>7</b>
5.1	File and Directory Layout . . . . .	7
<b>6</b>	<b>Security Considerations</b>	<b>8</b>
<b>7</b>	<b>IANA Considerations</b>	<b>9</b>
<b>8</b>	<b>XMPP Registrar Considerations</b>	<b>9</b>
8.1	Protocol Namespaces . . . . .	9
8.2	Protocol Versioning . . . . .	9
<b>9</b>	<b>XML Schema</b>	<b>9</b>

## 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](#)<sup>1</sup>.
- 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.

---

<sup>1</sup>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, and SHOULD have a 'password' attribute, which contains the user's password.

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' password='s3crEt'>
      [ ... ]
    </user>
  </host>
  <host jid='montague.net'>
    <user name='romeo' password='iLuvJuLi3T'>
      [ ... ]
    </user>
  </host>
</server-data>
```

## 4.3 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<sup>2</sup>.

Listing 4: The roster

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet' password='s3crEt'>
      <query xmlns='jabber:iq:roster'>
        <item jid='romeo@montague.net'
              name='Romeo'
              subscription='both'>
          <group>Friends</group>
        </item>
      </query>
    </user>
  </host>
</server-data>
```

<sup>2</sup>RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <<http://tools.ietf.org/html/rfc6121>>.

#### 4.4 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 5: Offline messages

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet' password='s3crEt'>
      <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>
```

#### 4.5 Private XML Storage

Private data stored by the server as specified in [Private XML Storage \(XEP-0049\)](#)<sup>3</sup> 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.

Listing 6: Private XML Storage

```
<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='shakespeare.lit'>
    <user name='hamlet' password='2b0Rnot2B'>
      <query xmlns="jabber:iq:private">
        <exodus xmlns="exodus:prefs">
          <defaultnick>Hamlet</defaultnick>
        </exodus>
      </query>
    </user>
  </host>
</server-data>
```

<sup>3</sup>XEP-0049: Private XML Storage <<https://xmpp.org/extensions/xep-0049.html>>.

```

        </exodus>
    </query>
</user>
</host>
</server-data>

```

## 4.6 vCards

By [vcard-temp \(XEP-0054\)](#)<sup>4</sup>, 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 7: vCards

```

<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet' password='s3crEt'>
      <vCard xmlns='vcard-temp'>
        <FN>Juliet Capulet</FN>
      </vCard>
    </user>
  </host>
</server-data>

```

## 4.7 Privacy Lists

Privacy lists, as specified in [Privacy Lists \(XEP-0016\)](#)<sup>5</sup>, 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 8: Privacy lists

```

<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet' password='s3crEt'>
      <query xmlns='jabber:iq:privacy'>
        <default name='public' />
        <list name='public'>
          <item type='jid'

```

<sup>4</sup>XEP-0054: vcard-temp <<https://xmpp.org/extensions/xep-0054.html>>.

<sup>5</sup>XEP-0016: Privacy Lists <<https://xmpp.org/extensions/xep-0016.html>>.



```

        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>

```

#### 4.8 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 9: Incoming subscription requests

```

<?xml version='1.0' encoding='UTF-8'?>
<server-data xmlns='urn:xmpp:pie:0'>
  <host jid='capulet.com'>
    <user name='juliet' password='s3crEt'>
      <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>

```

## 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 10: 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 11: 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 12: The user file

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

```
<user xmlns='urn:xmpp:pie:0'
  name='juliet' password='s3crEt'>
  <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. 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 \(IANA\)](#)<sup>6</sup>.

## 8 XMPP Registrar Considerations

### 8.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:pie:0

The [XMPP Registrar](#)<sup>7</sup> includes the foregoing namespace in its registry at [<https://xmpp.org/registrar/namespaces.html>](https://xmpp.org/registrar/namespaces.html), as described in Section 4 of [XMPP Registrar Function \(XEP-0053\)](#)<sup>8</sup>.

### 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 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>
```

---

<sup>6</sup>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/>.

<sup>7</sup>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/>.

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

```
<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>
```