XEP-0098: Enhanced Private XML Storage

Iain Shigeoka
mailto:iain@jivesoftware.com
xmpp:smirk@jabber.com

2003-06-25
Version 0.1

Status	Type	Short Name
Deferred	Standards Track	private-xml

Standardizes ”private” XML data storage.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

The 'jabber:iq:private' namespace has been documented in Private XML Storage (XEP-0049) [1] according to the historical behavior of current implementations. However there are two backward compatible improvements to the protocol introduced in this standard that increase the future useability of the protocol: matching on the fully qualified name of the XML fragment root, and the introduction of a standard mechanism of removing stored data. Because the protocol defined herein is not identical to 'jabber:iq:private', a new namespace name is used: 'http://jabber.org/protocol/private-xml'.
This protocol is designed to provide a simple interface to XML data storage on XMPP servers. The simple interface eases the implementation burden for the most basic data storage use-cases (e.g. storing simple client preferences on the server). More sophisticated XML data storage protocols should be built on top of, or compatible with this standard.

2 private-xml Namespace

2.1 Description

A Jabber client can store any arbitrary XML on the server side by sending an <iq/> chunk of type "set" to the server with a <query/> child scoped by the 'http://jabber.org/protocol/private-xml' namespace. The <query/> element MUST contain a single, arbitrary XML fragment. That fragment MUST be scoped by its own namespace. Any existing data stored on the server with the same fully qualified element name (tag name + namespace) is replaced by the new data.
The data can then be retrieved by sending an <iq/> of type "get" with a <query/> child scoped by the 'http://jabber.org/protocol/private-xml' namespace, which in turn MUST contain a single child element scoped by the namespace used for storage of that fragment. The fully qualified element name is used to locate matching XML data on the server. If no matching data is found, the server will respond with the empty query child element and not an error.
Finally, existing data on the server can be removed by sending an <iq/> of type "set" with a <query/> child scoped by the 'http://jabber.org/protocol/private-xml' namespace and containing an 'action' attribute with value 'delete', which in turn MUST contain a single child element scoped by the namespace used for storage of that fragment. The fully qualified element name is used to locate matching XML data on the server. The server responds with a successful result whether a matching data fragment was found or not (it's successful because the provided data no longer exits on the server). Deleting data using this method is indistinguishable from setting an empty XML fragment as far as the behavior this protoco is concerned. However, deleting data MUST remove the data from the server which may be implemented differently than the case of setting the data to an empty element. This may have significance in the context of future advanced XML storage protocols. Using the basic private XML data storage protocol, Jabber entities can create, read, update, and delete private data on the server. The data stored might be anything, as long as it is valid XML. One typical usage

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

for this namespace is the server-side storage of client preferences.

2.2 Methods

get	Sent with a blank query to retrieve the private data from the server.
set	Sent with the private XML data contained inside of a query.
set action='delete'	Sent with a blank query to delete private data from the server.
result	Returns the private data from the server.
error	There was an error processing the request. The exact error can be found in the child error element.

2.3 Elements

The root element of this namespace is query. A single child element with a proper namespace must be included otherwise the server will respond with error code 406. Only one element can be queried or set in a single IQ request. However, multiple elements, each containing data, can be stored independently on the server using separate set queries.

Listing 1: Client Stores Private Data

```
CLIENT:
<iq type="set" id="1001">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <exodus xmlns="exodus:prefs">
      <defaultnick>Hamlet</defaultnick>
    </exodus>
  </query>
</iq>

SERVER:
<iq
    type="result"
    from="hamlet@shakespeare.lit/denmark"
    to="hamlet@shakespeare.lit/denmark"
    id="1001"/>
```

Listing 2: Client Retrieves Private Data

```
CLIENT:
<iq type="get" id="1002">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <exodus xmlns="exodus:prefs"/>
  </query>
</iq>
```

```
SERVER:
<iq
    type="result"
    from="hamlet@shakespeare.lit/denmark"
    to="hamlet@shakespeare.lit/denmark"
    id="1002">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <exodus xmlns="exodus:prefs">
      <defaultnick>Hamlet</defaultnick>
    </exodus>
  </query>
</iq>
```

If a user attempts to get or set http://jabber.org/protocol/private-xml data that belongs to another user, the server must return an error to the sender. The error commonly used is 503 (Service Unavailable).

Listing 3: User Attempts to Get or Set Data for Another User

```
CLIENT:
<iq type="set" to="hamlet@shakespeare.lit" id="1003">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <exodus xmlns="exodus:prefs">
      <defaultnick>Macbeth</defaultnick>
    </exodus>
  </query>
</iq>

SERVER:
<iq
    type="error"
    from="hamlet@shakespeare.lit"
    to="macbeth@shakespeare.lit"
    iq="1003">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <exodus xmlns="exodus:prefs">
      <defaultnick>Macbeth</defaultnick>
    </exodus>
  </query>
  <error code="503">Service Unavailable</error>
</iq>
```

If a user attempts to perform a get without providing a child element, the server should return a 406 (Not Acceptable) error:

Listing 4: User Attempts to Get Data Without Specifying Child Element/Namespace

```
CLIENT:
```

```
<iq type="set" id="1004">
  <query xmlns="http://jabber.org/protocol/private-xml"/>
</iq>

SERVER:
<iq type="error" iq="1004">
  <query xmlns="http://jabber.org/protocol/private-xml"/>
  <error code="406">Not Acceptable</error>
</iq>
```

Certain namespaces are reserved in Jabber (namespaces beginning with 'jabber:' or 'http://jabber.org/', as well as 'vcard-temp'). If a user attempts to get or set http://jabber.org/protocol/private-xml data in a reserved namespace, historically some server implementations have chosen to return an error (commonly 406 [Not Acceptable]) to the sender. Such behavior is not required in order to comply with this document, but may be encountered by clients when interacting with some current server implementations.

Listing 5: User Attempts to Get or Set Data in a Reserved Namespace

```
CLIENT:
<iq type="set" id="1005">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <vCard xmlns="vcard-temp">
      <FN>Hamlet</FN>
    </vCard>
  </query>
</iq>

SERVER (optional error):
<iq type="error" iq="1005">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <vCard xmlns="vcard-temp">
      <FN>Hamlet</FN>
    </vCard>
  </query>
  <error code="406">Not Acceptable</error>
</iq>
```

The server always replies to a properly formatted get query with a result response rather than some form of 'not found' error. for exmple, the following shows the response from a server that does not have XML data under the 'data' name and 'imaginary' namespace.

Listing 6: User Attempts to Get Data in That Does Not Exist

```
CLIENT:
<iq type="get" id="1006">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <data xmlns="imaginary"/>
```

```
  </query>
</iq>

SERVER (does not have data in "imaginary" namespace, returns empty
    element):
<iq
    type="result"
    from="hamlet@shakespeare.lit/denmark"
    to="hamlet@shakespeare.lit/denmark"
    id="1006">
  <query xmlns="http://jabber.org/protocol/private-xml">
    <data xmlns="imaginary"/>
  </query>
</iq>
```

Finally, the client can delete data from the server using the delete query action.

Listing 7: User Deletes Data

```
CLIENT:
<iq type="get" id="1006">
  <query xmlns="http://jabber.org/protocol/private-xml" action="delete
      ">
    <exodus xmlns="exodus:prefs"/>
  </query>
</iq>

SERVER (server responds with success):
<iq
    type="result"
    from="hamlet@shakespeare.lit/denmark"
    to="hamlet@shakespeare.lit/denmark"
    id="1006"/>
```

2.4 Error Codes

Code	Text	Description
406	Not Acceptable	The IQ get does not contain a child element or (optionally) the IQ get or set is in a reserved namespace.
503	Service Unavailable	The IQ get or set is sent to a JID other than that of the sender.

2.5 DTD

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT query (#PCDATA)>
```