XEP-0284: Shared XML Editing

Joonas Govenius
mailto:joonas@uwc.net
xmpp:joonas@jabber.org

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

Tom Pusateri
mailto:pusateri@bangj.com

2021-03-04
Version 0.1.3

Status	Type	Short Name
Deferred	Standards Track	NOT YET ASSIGNED

This specification defines a protocol that enables two or more endpoints to collaboratively edit an XML object. The protocol is intended for use mainly over the Extensible Messaging and Presence Protocol (XMPP), either by existing instant messaging clients or by specialized editing clients. However, the protocol could also be used over a direct TCP connection rather than over XMPP.

Contents

1 Introduction

This specification defines a protocol for collaboratively editing XML data. Essentially, this protocol provides a simple way of synchronizing an unordered set of records across several endpoints. Additionally, this protocol defines a mapping between such a set of records and the Document Object Model (DOM).
A special feature of this protocol compared to most other collaborative editing tools is that no central or server entity is required. A Multi-User Chat (XEP-0045) [1] component or specialized editing component can be used for sessions that have a large number of participants, that need to be persistent, or that require more granular access control. However, the client implementation is minimally different whether such a specialized component is used or whether the session is one-to-one or multi-user.

2 Requirements

Requirements for shared editing are provided in Requirements for Shared Editing (XEP-0228) [2].

3 Glossary

GUID: a Globally Unique Indentifier, used as the identifier for a shared editing session.
Host: The JID to which the SXE messages are sent for relaying to other members of the session; this can be the initiator of the session (e.g., in a one-to-one session or small multi-user session) or a multi-user chat room or specialized shared editing component.
RID: the Record ID given to a record when it is created.
State: In the context of a new user joining, the state refers to the set of records that describes the edited object, including all previous versions of each record. All entities involved in the session are REQUIRED to keep this state unless a specialized component handles user joins.
Weight: Primarily, the weight of a node is represented by the 'primary-weight' field of the corresponding record. Secondarily, if the values of the 'primary-weight' of two records are equal, the first differing characters of the rids are compared by their Unicode values. The higher the character the higher the weight of the node is.

4 Session Management

When used in the context of XMPP, Shared XML Editing relies on Jingle (XEP-0166) [3] for overall session management. In Jingle terms, SXE defines a "transport method" that can be used for

[1] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[2] XEP-0228: Requirements for Shared Editing <https://xmpp.org/extensions/xep-0228.html>.
[3] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.

multiple "application types" such as XHTML documents, SVG whiteboards, Collaborative Data Objects (XEP-0204) [4], or any other XML data format.

4.1 Initiation

In order to initiate a shared editing session, one party sends a Jingle session-initiate request to another party. Here the <transport/> element indicates support for exchanging SXE information over XMPP and the <description/> element indicates support for editing of XHTML documents (this application type has not been defined yet and is used here only as an example). The <transport/> element MUST include at least one <host/> element that specifies the entity that serves as the host for the session. The host can be the initiator of the session (and must be for a one-to-one session) or a multi-user chat room or specialized reflector where the session is hosted.

Listing 1: Initiator sends session-initiate

```
<iq from='kingclaudius@shakespeare.lit/castle'
    id='jingle1'
    to='laertes@shakespeare.lit/castle'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='kingclaudius@shakespeare.lit/castle'
          sid='851ba2'>
    <content creator='initiator' name='this-is-the-editing-content'>
      <description xmlns='urn:xmpp:jingle:apps:xhtml'/>
      <transport xmlns='urn:xmpp:jingle:transports:sxe'>
        <host>kingclaudius@shakespeare.lit</host>
      </transport>
    </content>
  </jingle>
</iq>
```

The responder immediately acknowledges receipt of the session-initiate.

Listing 2: Responder acknowledges session-initiate

```
<iq from='laertes@shakespeare.lit/castle'
    id='jingle1'
    to='kingclaudius@shakespeare.lit/castle'
    type='result'/>
```

[4] XEP-0204: Collaborative Data Objects <https://xmpp.org/extensions/xep-0204.html>.

4.2 Session Acceptance

In order to definitively accept the session, the responder sends a session-accept to the initiator.

Listing 3: Responder sends session-accept

```
<iq from='laertes@shakespeare.lit/castle'
    id='jingle2'
    to='kingclaudius@shakespeare.lit/castle'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          initiator='kingclaudius@shakespeare.lit/castle'
          sid='851ba2'>
    <content creator='initiator' name='this-is-the-editing-content'>
      <description xmlns='urn:xmpp:jingle:apps:xhtml'/>
      <transport xmlns='urn:xmpp:jingle:transports:sxe'>
        <host>kingclaudius@shakespeare.lit</host>
      </transport>
    </content>
  </jingle>
</iq>
```

The initiator immediately acknowledges receipt of the session-accept.

Listing 4: Initiator acknowledges session-accept

```
<iq from='kingclaudius@shakespeare.lit/castle'
    id='jingle2'
    to='laertes@shakespeare.lit/castle'
    type='result'/>
```

4.3 Session Refusal

In order to decline the session, the responder sends a session-terminate to the initiator.

Listing 5: Responder sends session-terminate

```
<iq from='laertes@shakespeare.lit/castle'
    id='jingle3'
    to='kingclaudius@shakespeare.lit/castle'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          host='chamber@conference.shakespeare.lit'
          initiator='kingclaudius@shakespeare.lit/castle'
          sid='851ba2'>
    <reason>
```

```
      <unsupported-applications/>
    </reason>
  </jingle>
</iq>
```

The responder indiciates the reason for refusing the session by including a "reason" element. The following reasons are suggested (see also Jingle (XEP-0166) [5]):

- alternative-session -- the responder already has an active session with the initiator and wishes to use that session instead.
- decline -- the responder formally declines the session.
- unsupported-applications -- the responder supports none of the offered the application types.

The initiator immediately acknowledges receipt of the session-terminate.

Listing 6: Initiator acknowledges session-terminate

```
<iq from='kingclaudius@shakespeare.lit/castle'
    id='jingle3'
    to='laertes@shakespeare.lit/castle'
    type='result'/>
```

5 Determining the features of the Host

Before connecting to a session, a party MUST determine the Service Discovery (XEP-0030) [6] identity and supported features of the host. In many situations the party already knows this information (e.g., if the host is the initiator of the Jingle session). In other situations the party will need to send a service discovery information request to the host.

Listing 7: Service Discovery Request

```
<iq from='laertes@shakespeare.lit/castle'
    to='kingclaudius@shakespeare.lit/castle'
    id='disco1'>
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

If the host supports Shared XML Editing over XMPP, it MUST return features of "urn:xmpp:sxe:0" and "urn:xmpp:jingle:transports:sxe" (see Protocol Namespaces regarding

[5]XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[6]XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

issuance of one or more permanent namespaces):

Listing 8: Service discovery response

```
<iq from='laertes@shakespeare.lit/castle' to='kingclaudius@shakespeare
    .lit/castle' type='result' id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='client' type='pc'/>
    ...
    <feature var='urn:xmpp:sxe:0'/>
    <feature var='urn:xmpp:jingle:transports:sxe'/>
    ...
  </query>
</iq>
```

6 Advertising a Session

An entity that is engaged in a session can advertise that fact by including the session id and descriptive name in its XMPP presence.

Listing 9: Advertising a session

```
<presence from='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0' session='851ba2' name='Marketing␣
      Materials'/>
</presence>
```

Such presence can be broadcast or can be sent in the context of a multi-user chat room (see Multi-User Chat (XEP-0045) [7]).
However, presence is not sent when operating in a serverless messaging environment (see Link-Local Messaging (XEP-0174) [8]). Instead, DNS TXT records are published. Two new key-value pairs are used to advertise a session id (”sxe_id”) and session name (”sxe_name”) when using serverless messaging.

7 Connecting to a Session

In order to synchronize its local state with the existing state of the edited object, an entity sends a connection request to the host. The identity learned through service discovery determines what kind of message the joining party sends (e.g., type='groupchat' if the host is a MUC room).
To connect to a session, the joiner MUST send an SXE <connect/> element to the host.

[7]XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[8]XEP-0174: Link-Local Messaging <https://xmpp.org/extensions/xep-0174.html>.

Listing 10: Connecting to a session

```
<message
   from='laertes@shakespeare.lit/castle'
   to='kingclaudius@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0' id='e' session='851ba2'>
    <connect/>
  </sxe>
</message>
```

8 Initial State Synchronization

8.1 Making a State Offer

When a joining user sends a connect request, the joiner must retrieve the state of the session from an existing participant. The following ruules apply:

1. In a one-to-one session, the initiator MUST offer to send the state.
2. In a multi-user session, both the host and the invitor (initiator) MUST offer to send the state. However, other participants MAY offer to send the state. The joiner SHOULD prefer to receive the state from the host or initiator.

The state offer MUST contain the <description/> element or elements that were included in the Jingle session-initiate request.

Listing 11: A state offer

```
<message
  from='kingclaudius@shakespeare.lit/castle'
  to='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='f'>
    <state-offer>
      <description xmlns='urn:xmpp:jingle:apps:xhtml'/>
    </state-offer>
  </sxe>
</message>
```

8.2 Accepting a State Offer

The joiner accepts a state offer by sending an <accept-state/> element to one of the entities that offer to send the state. The joiner MUST store all the <sxe/> elements it receives after the <state-offer/> element it decides to accept. It MUST also abort the negotiation with the other users that offered to send the state.

Once the other entity receives the <accept-state/> element it MUST proceed sending the state as described in the next section.

Listing 12: Accepting a state offer

```
<message
   from='laertes@shakespeare.lit/castle'
   to='kingclaudius@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='g'>
    <accept-state/>
  </sxe>
</message>
```

8.3 Refusing a State Offer

If a multiple state offers were received, one should be accepted and the others should be refused by sending a <refuse-state/> element.

Listing 13: Refusing a state offer

```
<message
   from='laertes@shakespeare.lit/castle'
   to='kingclaudius@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='g'>
    <refuse-state/>
  </sxe>
</message>
```

8.4 Sending the State of the Document

The process of sending the state is following:

1. The sender sends a <document-begin/> element and simultaneously takes a "snapshot" of the document's state. The <document-begin/> element SHOULD have a 'prolog' attribute. The 'prolog' attribute should contain the XML prolog of the document encoded using the data: URI scheme (RFC 2397).

2. The sender sends the state as it was at the time of sending <document-begin/>.

3. The sender sends a <document-end/> element.

The state can be sent in any number of <sxe/> elements but the user sending the state MUST NOT send any new <sxe/> elements between sending the <document-begin/> (i.e. taking the snapshot) and the <document-end/> element.
The state SHOULD include a version of each element that was synchronized, and hence won't be undone, as well as all the later versions. In practice this can often be impossible to know in a session without a specialized MUC component so it may be safest to send version 0 and all the later edits to each element. Version 0 is implied if the version element is missing.

Listing 14: Sending the state of a blank document (only prolog)

```
<message
  from='kingclaudius@shakespeare.lit/castle'
  to='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='b'>
    <state>
      <document-begin
          prolog='data:text/xml,%3C%3Fxml%20version%3D%271.\
0%27%20standalone%3D%27no%27%3F%3E%0A%3C%21DOCTYPE%20svg%\
20PUBLIC%20%27-%2F%2FW3C%2F%2FDTD%20SVG%201.1%2F%2FEN%27%\
20%27http%3A%2F%2Fwww.w3.org%2FGraphics%2FSVG%2F1.1%2FDTD\
%2Fsvg11.dtd%27%3E%0A'/>
      <document-end last-sender='{}' last-id='{}' />
    </state>
  </sxe>
</message>
```

If the session is already in progress, the entity sends the snapshot.

Listing 15: Sending the state of an existing document

```
<message
  from='kingclaudius@shakespeare.lit/castle'
  to='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='bxyz'>
    <state>
      <document-begin
        prolog='data:text/xml,%3C!DOCTYPE%20html%0D%0APUBLIC%20%22-\
␣␣%2F%2FW3C%2F%2FDTD%20XHTML%201.0%20Strict%2F%2FEN%22%0D%0A%22htt\
␣␣p%3A%2F%2Fwww.w3.org%2FTR%2Fxhtml1%2FDTD%2Fxhtml1-strict.dtd%223\
␣␣E%0D%0A'/>
      <new type='processinginstruction'
           pitarget='xml-stylesheet'
           pidata='href="style.xsl"␣type="text/xsl"'
           rid='GUID0'
           primary-weight='0.4' />
      <new type='element'
           ns='http://www.w3.org/1999/xhtml'
           name='html'
           rid='GUID1'
           primary-weight='3' />
      <new type='attr'
           ns='xml'
           name='lang'
           chdata='fi'
           parent='GUID1'
           rid='GUID2' />
      <set target='GUID2'
           version='1'
           chdata='en' />
      <new type='element'
           name='head'
           parent='GUID1'
           rid='GUID3' />
      <new type='element'
           name='title'
           parent='GUID3'
           rid='GUID4' />
      <new type='text'
           chdata='Royal␣Musings'
           rid='GUID5'
           parent='GUID4'
           primary-weight='0'/>
      <new type='comment'
           chdata='The␣title␣of␣the␣document␣goes␣here.'
           rid='GUID6'
           parent='GUID4'
           primary-weight='-2.43'/>
      <new type='element'
           name='body'
           parent='GUID1'
           rid='GUID7'
           primary-weight='23' />
      <document-end
          last-sender='jid3'
          last-id='abc'/>
    </state>
  </sxe>
</message>
```

9 Subsequent State Changes

Once a participant has received initial state, he can participate in the editing session.

Listing 16: An edit

```
<message
    to='laertes@shakespeare.lit/castle'
    from='kingclaudius@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
       session='851ba2'
       id='4'>
    <state>
      <new type='element'
           name='p'
           parent='GUID7'
           rid='GUID8' />
      <new type='text'
           chdata='It&apos;s␣good␣to␣be␣the␣king!'
           parent='GUID8'
           rid='GUID9'/>
      <set target='GUID5'
           version='1'
           chdata='It&apos;s␣good␣to␣be␣the␣king!'/>
    </state>
  </sxe>
</message>
```

Here is another edit in the session.

Listing 17: Another edit

```
<message
    from='kingclaudius@shakespeare.lit/castle'
    to='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='4'>
    <state>
      <new type='element'
           name='p'
           parent='GUID7'
           rid='GUID10' />
      <new type='text'
           chdata='It␣certainly␣is!'
           parent='GUID10'
           rid='GUID11' />
      <set target='GUID5'
           version='1'
```

```
          chdata='It␣certainly␣is' />
      <set target='GUID6'
          version='1'
          replacefrom='9'
          replacen='16'
          chdata='{}' />
      <set target='GUID6'
          version='2'
          replacefrom='3'
          replacen='0'
          chdata='document'/>
    </state>
  </sxe>
</message>
```

This basic editing session results in the following XML.

Listing 18: Resulting XHTML document (with additional whitespace nodes for clarity)

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD␣XHTML␣1.0␣Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<?xml-stylesheet href="style.xsl" type="text/xsl"?>
<html>
  <head>
    <!-{}-The document title goes here.-{}->
    <title>Royal Musings</title>
  </head>
  <body>
    <p>It&apos;s good to be the king!</p>
    <p>It certainly is!</p>
  </body>
</html>
```

10 Formal Definition

10.1 SXE Element

All SXE elements MUST be contained in a <sxe/> element. This element MUST posess the following attributes:

Attribute	Description
xmlns	REQUIRED and MUST be "urn:xmpp:sxe:0"
session	REQUIRED and MUST be a GUID of the session.
id	REQUIRED and MUST be unique within the set of <sxe/> elements sent by the user to the session.

10.2 XML Document Requirements

The prolog of the XML document cannot be edited after the session has been established.
If an XML Schema Definition is specified for the document and the processing of an <sxe/> element results in a noncompliant document, the receiving client SHOULD reply with edits that effectively undo the offending edits. TODO: the offending client should probably be notified.

10.3 Mapping the Records to the DOM Document

A record contains the following fields and corresponds to a single DOM node:

Field Name	Mutable	Applies to nodes of type	Description
rid	no	all	The GUID of the record.
type	no	all	The type of DOM node (element, attr, etc.).
version	Indirectly	all	The current version of the record.
parent	yes	all	The record id of the record corresponding to the parent node.
primary-weight	yes	all	The primary weight used to determine the order of sibling nodes corresponding to the records.
ns	yes (TODO: is this reasonable?)	element, attr	The namespace of the element or attribute
name	yes (TODO: is this reasonable?)	element, attr	The name of the element or attribute.
chdata	yes	text, attr, comment	The content of a text node or a comment or the value of the attribute.
pitarget	yes (TODO: is this reasonable?)	proccessinginstruction	The target of the processing instruction.
pidata	yes	proccessinginstruction	The data of the processing instruction.

Whenever a record is added, modified, or removed, the client MUST ensure that the DOM nodes corresponding to the records meet the following criteria:

1. The parent of each DOM node MUST be the node corresponding to the record specified by the 'record id' field of the record of the node. If no such record exists, the "orphan" record MUST be deleted. If the 'parent' field is empty, the node corresponding to the record MUST be a child of the DOM document itself. If two root nodes exist, the record with lower secondary weight MUST be removed.
2. Each node MUST be located after the child node that has the greatest weight less than the weight of the node.
3. The namespace of each DOM element and attribute MUST be equal to the 'name' field of the corresponding record.
4. The name of each DOM element and attribute MUST be equal to the 'name' field of the corresponding record. If two records for attribute nodes specify the same 'name' and 'parent', the record with lower secondary weight MUST be removed.
5. The value of each DOM attribute node MUST be equal to the 'chdata' field of the corresponding record.
6. The content of each DOM text node MUST be equal to the 'chdata' field of the corresponding record.
7. The content of each DOM comment node MUST be equal to the 'chdata' field of the corresponding record.
8. The target of each DOM processing instruction MUST be equal to the 'pitarget' field of the corresponding record.
9. The data of each DOM processing instruction MUST be equal to the 'pidata' field of the corresponding record.
10. The DOM nodes do not offend the XML Schema. If they do, the client SHOULD send edits to correct the situation.

rid	type	version	parent	primary-weight	ns	name	chdata	pitarget	pidata
GUID0	element	0		0	http://www.w3.org/2000/svg	svg	N/A	N/A	N/A
GUID1	element	0	GUID0	0		path	N/A	N/A	N/A
GUID2	attr	0	GUID1	0		d	M10 10L20 20L20 10Z	N/A	N/A
GUID9	element	0	GUID0	3.4		g	N/A	N/A	N/A

rid	type	version	parent	primary-weight	ns	name	chdata	pitarget	pidata
GUID5	element	0	GUID0	3.4		circle	N/A	N/A	N/A
GUID6	attr	3	GUID5	0		cx	10	N/A	N/A
GUID7	attr	1	GUID5	1		cy	20	N/A	N/A
GUID8	attr	0	GUID5	2		r	5	N/A	N/A

Listing 19: Corresponding XML document (without the XML prolog)

```
<svg xmlns='http://www.w3.org/2000/svg'>
  <path d='M10␣10L20␣20L20␣10Z' />
  <circle cx='10' cy='20' r='5' />
  <g />
</svg>
```

10.4 Commutative and Non-commutative Edits

Changes to the records can be divided into two categories: commutative and non-commutative edits.
Commutative edits are commutative with all edits and are never "undone" so keeping a history of them is NOT REQUIRED. Edits that add or remove records are commutative.
An edit that changes an existing record is called non-commutative because it may not be commutative with edits that change the same record. Hence these changes may need to be undone so keeping a history of the changes caused by such edits is REQUIRED. The breadth of this required history depends on the role of the entity and on whether the session works through a server component:

	By server	By client
Server exists	All non-commutative edits, that have not been undone to records that have not been removed; must store the creator and last modifier of each node (to be included as 'creator' and 'last-modified-by' attributes in <new/> elements sent to joining clients).	Non-commutative edits sent by the client itself. May be removed once a further non-commutative edits to the same record from another entity is received.
Server does not exists	---	All non-commutative edits, that have not been undone, to records that have not been removed.

10.4.1 Requirements for the Server Component

The server MUST apply commutatative edits to its local copy like a client and pass on the edits without changes.
The server component intercepts and modifies non-commutative edits in order to reduce the history requirements of the clients as indicated above. Once it receives a non-commutative edit, it MUST take the following action depending on whether the version number of the edit is "in-order" (one higher than the current version) or "out-of-order":

1. The server receives an in-order non-commutative edit: the server does not modify the edit, applies it locally, and passes it on normally.
2. The server receives an out-of-order non-commutative edit: it processes the edit like a client would in order to locally undo conflicting edits; then, instead of passing on the out-of-order edit, the server replaces the edit by an in-order non-commutative edit that results in a record identical to what the server has locally after the (possible) undos. Note that this edit may be a "no-op" that merely increases the version of the target.

10.5 Commutative Edits

10.5.1 Creating New Nodes

A client can add a new node to the document by adding a record with the <new/> element.

Attribute	Description
rid	REQUIRED. MUST be a GUID.
version	OPTIONAL. MUST be a non-negative integer. Assumed to be 0 if not present.
parent	REQUIRED (Except at top level.)
primary-weight	OPTIONAL. MUST be a float. Assumed to be 0 if not present.
type	REQUIRED. MUST be 'element', 'attr', 'text', 'comment', or 'processinginstruction'
ns	OPTIONAL if 'type' is 'element' or 'attr'. Not allowed otherwise.
name	REQUIRED if 'type' is 'element' or 'attr'. Not allowed otherwise.
chdata	REQUIRED if 'type' is 'attr', 'text', or 'comment'. Not allowed otherwise.
pitarget	REQUIRED if 'type' is 'processinginstruction'. Not allowed otherwise.
pidata	REQUIRED if 'type' is 'processinginstruction'. Not allowed otherwise.
creator	OPTIONAL. MUST be the JID or room nick of the creator of the node.
last-modified-by	OPTIONAL. MUST be the JID or room nick of the user who last modified the node.

Listing 20: Sending new nodes

```
<message
  from='kingclaudius@shakespeare.lit/castle'
  to='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='11'>
    <new type='element'
          name='path'
          parent='GUID1'
          rid='GUID4' />
    <new type='attr'
          name='d'
          parent='GUID4'
          rid='GUID5'
          chdata='M10␣10L30␣50L50␣10Z' />
  </sxe>
</message>
```

To process a <new/> element the client MUST create a new record with the values of the attributes stored in the corresponding fields.

10.5.2 Removing Nodes

A client can remove any node in the document by removing the corresponding record with the <remove/> element.

Attribute	Description
target	REQUIRED and MUST be the record id of the node to be removed.

A client MUST NOT send a <remove/> element that removes a node that has child nodes without explicitly removing the records of those nodes first.

Listing 21: Removing an existing nodes.

```
<message
  from='kingclaudius@shakespeare.lit/castle'
  to='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='13'>
    <remove target='GUID5'/>
    <remove target='GUID4'/>
  </sxe>
</message>
```

To processes a <remove/> element the client MUST remove the record specified by the 'target' attribute.
If the node corresponding to the target record has child nodes, the receiver MUST send <remove/> elements for each of them as described above.

10.6 Non-commutative Edits

10.6.1 The set Element

The <set/> element is used to modify an existing record.

Attribute	Description
target	REQUIRED and MUST be the record id of the node being modified.
version	REQUIRED and MUST be the current version of the node incremented by one.
parent	OPTIONAL.
primary-weight	OPTIONAL.
ns	OPTIONAL but only allowed if the target record is of type 'element' or 'attr'.
name	OPTIONAL but only allowed if the target record is of type 'element' or 'attr'.
chdata	OPTIONAL but only allowed if the target node is of type 'attr', 'text', or 'comment'.
pitarget	OPTIONAL but only allowed if the target record is of type 'processinginstruction'.
pidata	OPTIONAL but only allowed if the target record is of type 'processinginstruction'.
replacefrom	replace from position. OPTIONAL but only allowed if 'chdata' and 'replacen' attributes are also included. MUST be a non-negative integer.
replacen	replace n characters. OPTIONAL but only allowed if 'chdata' and 'replacefrom' attributes are also included. MUST be a non-negative integer.

Listing 22: set elements.

```
<message
  from='kingclaudius@shakespeare.lit/castle'
  to='laertes@shakespeare.lit/castle'>
  <sxe xmlns='urn:xmpp:sxe:0'
      session='851ba2'
      id='14'>
```

```
    <set target='GUID14'
      version='1'
      chdata='10' />

    <set target='GUID8'
      version='1'
      parent='GUID1'
      primary-weight='8' />

  </sxe>
</message>
```

10.6.2 Processing a set Element

To processes a <set/> element the client MUST follow the following steps:

1. If the record specified by the 'target' attribute doesn't exist, the <set/> element MUST be ignored.
2. If the client is receiving history, (i.e. edits sent between the <document-begin/> and <document-end/> elements), it MUST set the version of the target record to the value of the 'version' attribute. Otherwise, the client MUST increment the version of the target recordrecord by one
3. If the version of of the target record is now equal to the 'version' attribute of the <set/> element, the client MUST store the values of the attributes of the <set/> element as updated values of the corresponding fields in the record. The only exception occurs when 'chdata', 'replacefrom', and 'replacen' are specified: in that case, n characters starting from the f'th character of the existing 'chdata' field MUST be replaced by c, where n, f, and c are the values of the 'replacen', 'replacefrom', and 'chdata' attributes respectively.
   Otherwise all fields of the record, except for the 'version' field, must be reverted to version (v - 1), where v is the value of the 'version' attribute of the received <set/> element.

11 Implementation Notes

11.1 MUC Roles

It should be noted that given the MUC specification and the requirement of this protocol to send a connect request message to all room members in order to join an existing session, you effectively can not use the visitor role of MUC with a regular MUC server component. However, the specialized MUC component, if used, MUST accept and respond to the connection requests even if they come from users who do not have voice in the room.

12 Security Considerations

To follow.

13 IANA Considerations

This XEP requires no interaction with the Internet Assigned Numbers Authority (IANA) [9].

14 XMPP Registrar Considerations

14.1 Protocol Namespaces

Until this specification advances to a status of Draft, its associated namespaces shall be:

- urn:xmpp:sxe:0
- urn:xmpp:jingle:transports:sxe

Upon advancement of this specification, the XMPP Registrar [10] shall issue permanent namespaces in accordance with the process defined in Section 4 of XMPP Registrar Function (XEP-0053) [11].
The following namespaces are requested, and are thought to be unique per the XMPP Registrar's requirements:

- urn:xmpp:sxe
- urn:xmpp:jingle:transport:sxe

14.2 Service Discovery Identities

The &REGISTRAR; shall add the type of "sxe" to the "collaboration" category in its registry of service discovery identities.

[9]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/>.

[10]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/>.

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

14.3 Jingle Transport Methods

The XMPP Registrar shall include "sxe" in its registry of Jingle transport methods. The registry submission is as follows:

```
<transport>
  <name>sxe</name>
  <desc>A method for exchanging Shared XML Editing data over XMPP.</
     desc>
  <type>reliable</type>
  <doc>XEP-xxxx</doc>
</transport>
```

15 XML Schema

To follow.

16 Acknowledgements

The theory behind the synchronization of an individual entity (a record) was put forward by Mats Bengtsson (http://coccinella.sourceforge.net/docs/MemoSyncSVG-XMPP.txt). He also provided other input that helped form this XEP. Thanks to Dmitriy Chervov for his feedback based on implementation experience.