XEP-0113: Simple Whiteboarding

Huib-Jan Imbens
mailto:jabber@imbens.nl
xmpp:imbens@jabber.org

2003-09-07
Version 0.2

Status	Type	Short Name
Deferred	Informational	Not yet assigned

A proposal for an extremely simple whiteboarding protocol over Jabber.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

As explained in the now obsolete XEP-0010: Whiteboarding [1]: "Jabber is often thought of simply as a system for instant messaging, albeit an open one. However, Jabber technology can be used, and is being used, in applications quite different from simple IM. One of these applications is whiteboarding. In collaborative work, the ability to draw (for example, to design sketches, UML schemas, house architectures, and organizational plans) is essential, as exemplified by the success of real-world whiteboarding applications such as Microsoft NetMeeting. Whiteboarding can also be used for entertainment purposes such as games and quizzes. Because of the value of whiteboarding as an important real-time collaboration tool, other IM services are beginning to offer these capabilities. For these and other reasons, I believe that a good protocol for whiteboarding in Jabber would be of great value".
The increasing penetration of pen-based devices, such as PDAs and tablet PCs, makes the need for a protocol that allows for sending freehand drawing information more urgent.
Several attempts have been made to create a whiteboarding protocol for Jabber:

1. Collaborative Imaging (Whiteboarding via Streaming XPM) describes a protocol that sends partial bitmaps. This protocol is not suitable for freehand drawing and has not been implemented.
2. Jabber Whiteboarding using SVG [2] describes a protocol that uses a subset of SVG. It refers to a missing DTD that describes the precise subset, but there is little doubt that that subset will be hard to implement. This protocol has not been implemented.
3. The Coccinella client includes an open source implementation of a whiteboarding protocol. However, the protocol has not been documented and does not seem easy to implement. In fact it is mostly raw TCL, making an implementation of that protocol in a language other than TCL rather difficult.
4. The Tkabber client has a whiteboard plugin. The protocol has not been documented, but it uses a subset of SVG, similar to the one defined in this document.

2 Requirements

The protocol has the following requirements in order of importance:

1. It should allow for freehand drawing because that will be its principal use on pen-based devices.
2. It should be extremely easy to implement to ensure its rapid adaptation.
3. It should be light-weight.

[1] XEP-0010: Whiteboarding SIG http://www.xmpp.org/extensions/xep-0010.html
[2] Jabber Whiteboarding using SVG http://www.protocol7.com/jabber/whiteboard_proposal.txt

4. It should not require server modifications.

The following are definitely not objectives of the protocol:

1. It need not be complete. Eventually an SVG-based protocol will be defined that will either replace or coexist with this protocol and that will satisfy all drawing needs. However, given the history of whiteboarding protocols, such a protocol is far away.
2. It need not be extensible. As a ”Simple Whiteboarding protocol” it should not try to grow into a more complex protocol that would be more difficult to implement.

3 Use Cases

There are three scenarios where whiteboarding can be used:

- One person sends a single, completed, whiteboard to another person.
- The more typical scenario is the one where one person starts a whiteboard session with another person and both persons collaborate in the drawing. Both sides may add paths, move them around or delete them.
- Finally multiple people gathered in a conference room can use single whiteboard.

3.1 Single whiteboard message

Typically the user right-clicks on the destination contact and will select a ”whiteboard message” option. The client will show a dialog where the user can create the drawing. It is up to the implementation to decide whether the user can include text in the message as well. Upon clicking a send button the client will close the dialog and send the following message:

Listing 1: Single whiteboard message

```
<message
    from='painter@shakespeare.lit'
    to='timon@shakespeare.lit/hall'>
  <body>A piece of painting, which I do beseech your lordship to
     accept.</body>
  <x xmlns='http://jabber.org/protocol/swb'>
    <path d='M 100 100 L 300 100 200 300 100 100' stroke='#ff0000'
       stroke-width='1' id='painter1'/>
  </x>
</message>
```

The path node is a simplified SVG path node that allows only ’M’, ’m’, ’L’ and ’l’ commands. ’M’ (’m’) command is a (relative) moveto command, ’L’ (’l’) is a (relative) lineto command. All

four commands take one or more coordinate pairs (in pixels). 'M' sets the current point to the coordinate pair. 'm' adds the coordinate pair to the current point. 'L' draws a line from the current point to the point designated by the coordinate pair and sets the current point to the coordinate pair. 'l' draws a line from the current point to the sum of the currentpoint and the coordinate pair and adds the coordinate pair to the current point. The optional stroke attribute indicates the color of the path and defaults to black, the optional stroke-width indicates the width of the path in pixels and defaults to 1. The id attribute can be used for later reference to the path. If there is no id attribute, the path can not be referred to.
The path in example 1, draws a red triangle with vertices (100,100), (300,100) and (200, 300)
Other respresentations of the same path are 'M100.0,100.0L300.0,100.0,200.0,300.0,100.0,100.0', 'M100,100l200,0-100,200-100-200' and 'M100,100l200,0L200,300,100,100'. Note that in the second representation some commas can be left out because the sign indicates that a new coordinate is starting. This fact can be used to reduce data size as much as possible to avoid karma problems. A precise grammar of the "d" attribute is given below.
A typical implementation will generate such paths by adding an 'M' command with the mouse coordinates on a mouse down event and adding an 'L' command with the mouse coordinates on every mouse move event as long as the mouse is down. It is up to the implementation to decide whether to complete and send the message on a mouse up event or to wait for a click on a send button.

3.2 Whiteboard chat session

A more typical use case is where two clients share a whiteboard. Again the user will right click on the destination and will select a "whiteboard chat" option. The client will present a dialog where the user can create a drawing. Upon clicking a send button or releasing the mouse button, the client will send the following message:

Listing 2: Initiating a whiteboard chat session

```
<message
    from='kingclaudius@shakespeare.lit/castle'
    to='laertes@shakespeare.lit/castle'
    type='chat'>
  <thread>c357e044c676cc5e3c729d07544c87b58a366dba</thread>
  <body>... like the painting ... ?</body>
  <x xmlns='http://jabber.org/protocol/swb'>
    <path d='M100.0,100.0L300.0,100.0,200.0,300.0,100.0,100.0' id='
       kingclaudius1' />
  </x>
</message>
```

In this case the dialog will not close. At the destination client a similar dialog will pop up, allowing the user at the other end to add her own part of the drawing. The resulting message will look like this (line breaks provided for readability only):

Listing 3: Continuing a whiteboard chat session

```
<message
    from='laertes@shakespeare.lit/castle'
    to='kingclaudius@shakespeare.lit/castle'
    type='chat'>
  <thread>c357e044c676cc5e3c729d07544c87b58a366dba</thread>
  <x xmlns='http://jabber.org/protocol/swb'>
    <path d='M 32 41 L 33 40 33 39 34 38 34 37 35 36 35 34 36 33 37
             32 38 31 38 30 39 30 40 28 41 27 42 26 43 26 44 25 45
             24 46 24 48 23 50 22 52 21 53 21 54 21 55 21 55 20 56
             20 58 20 59 20 60 20 61 20 62 20 63 20 64 20 65 20 66
             20 67 20 68 20 69 20 69 21 70 21 71 22 72 23 72 24 73
             25 73 26 73 27 73 28 73 29 73 30 74 30 74 31 74 32 75
             33 75 34 75 35 75 36 75 37 75 38 75 39 75 40 75 41 75
             43 75 44 75 46 75 47 75 48 75 49 75 50 74 52 74 53 74
             54 73 55 72 55 72 57 72 58 71 58 70 60 69 61 69 63 68
             64 67 64 67 65 67 66 66 67 65 67 65 69 64 70 64 71 63
             72 62 73 62 74 62 75 61 75 60 76 60 77 59 77 59 78 59
             79 58 79 58 80 58 81 58 82 57 82 57 83 57 84 57 86 57
             87 56 87 56 88 56 89 55 89 55 90 55 91 55 92 54 93 54
             94 54 95 54 96 M 55 113 L 54 113 53 113 52 113 51 113
             49 114 49 115 48 115 47 115 47 116 47 117 46 117 45 117
             45 118 45 120 45 121 45 123 45 124 45 125 45 127 45 128
             45 130 46 131 46 132 46 133 47 133 47 134 48 134 49 134
             49 135 50 135 51 135 52 135 52 136 54 136 55 136 56 136
             57 136 58 136 59 136 59 135 60 134 61 133 61 132 61 131
             61 130 62 130 62 129 62 128 62 127 62 126 62 125 62 123
             62 122 62 120 61 120 61 119 61 118 61 117 60 117 59 117
             58 117 56 117 55 117 54 117' />
  </x>
</message>
```

It is left as a mental exercise to the reader to imagine Laertes answer. Alternatively the reader could build this protocol into her favorite Jabber client, set a breakpoint, and paste the path above at the appropriate place.
Alternatively Laertes could respond like:

Listing 4: Moving a path

```
<message
    from='laertes@shakespeare.lit/castle'
    to='kingclaudius@shakespeare.lit/castle'
    type='chat'>
  <thread>c357e044c676cc5e3c729d07544c87b58a366dba</thread>
  <x xmlns='http://jabber.org/protocol/swb'>
    <move id='kingclaudius1' dx='-100' dy='-100'/>
  </x>
</message>
```

This would move the King's triangle 100 pixels to the left and top, to the upper left corner of the screen.
If Laertes were bold enough he might even answer:

Listing 5: Deleting a path

```
<message
    from='laertes@shakespeare.lit/castle'
    to='kingclaudius@shakespeare.lit/castle'
    type='chat'>
  <thread>c357e044c676cc5e3c729d07544c87b58a366dba</thread>
  <x xmlns='http://jabber.org/protocol/swb'>
    <delete id='kingclaudius1'/>
  </x>
</message>
```

This would remove King Claudius's triangle from the screen.

3.3 Conference room whiteboard

The final use case is the one where multiple users, gathered in a conference room, share a single whiteboard. Messages will typically look like this:

Listing 6: Conference room whiteboard

```
<message
    from='nestor@shakespeare.lit'
    to='plains@conference.shakespeare.lit'
    type='groupchat'>
  <body>So, so, we draw together.</body>
  <x xmlns='http://jabber.org/protocol/swb'>
    <path d='M100,100l200,0L200,300,100,100' />
  </x>
</message>
```

4 Implementation Notes

4.1 The GUI

Usually when a user wants to send a message to a contact, the client will present her with a choice between sending a message or starting a chat. If the client implements the present protocol, the client can add the options of sending a whiteboard message and starting a whiteboard chat. Whether the client offers these options for an individual contact could be

based on standard Service Discovery (XEP-0030) [3] or Jabber Browsing (XEP-0011) [4] techniques. Presentation of a path in case of a "Single whiteboard message" is rather obvious. The presentation of multiple-user whiteboards, either chat or conference, leaves more to the imagination of the implementor. The implementor could decide to use different colors for paths drawn by different users. The saturation of a path could decrease with age.

4.2 Karma

One issue that will hinder all whiteboard protocol implementations is the karma problem. At least jabberd uses karma to make sure that a client does not send to much data to the server. This should help against denial-of-service attacks. When you use up all your karma, the server stops handling your messages for a while. This is a problem for whiteboards because it is much easier to send a lot of drawing data, than to send a lot of textual data. Usually combining paths, that is, sending paths when the user clicks on a send button instead of on mouse up, reduces data size because it reduces the overhead of the message element. Using the relative lineto command ('l') instead of the absolute lineto ('L') command will also reduce message size, because usually relative coordinates will only use one or two digits whereas absolute coordinates will typically use three. Finally implementations can reduce message size by not recording every mouse move event, e.g. by dropping mouse events whose locations would be accurately interpolated.

4.3 Text

The protocol does not provide explicit support for drawing text. The reason for this is that explicit support, eg. in the form of the SVG text element [5], would break the second and third requirements above. However a client can still provide text support by representing characters as paths, eg. by using a Hershey font.
The code snippet below shows the lines along which this could be done:

Listing 7: Coding the letter A into a path

```
// generating the path <path d='M14␣6l-8,21M14␣6l8,21M9␣20l10,0'/>
   from the letter 'A'

        static char* sHersheyFontData[] = {
                ...
                "I[RFJ[␣RRFZ[␣RMTWT", // the character A, consisting
                    of three strokes
                ...
                };
```

[3] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[4] XEP-0011: Jabber Browsing <https://xmpp.org/extensions/xep-0011.html>.
[5] Text - SVG 1.0 http://www.w3.org/TR/SVG/text.html

```
        for (int i = 0 ; sHersheyFontData ['A'][2*i+2] != 0 ; i++) {
                // read a new coordinate pair
                POINT myPoint = {sHersheyFontData ['A'][2*i+2]-'R',
                    sHersheyFontData ['A'][2*i+2+1]-'R')};
                // test for the special case pen up
                if (myPoint.x  == -50 && myPoint.y == 0) {
                        penUp = true;
                } else {
                        if (penUp) {
                                penUp = false;
                                currentPathSet.push_back (std::vector
                                    <POINT> ()); // pen goes down, add
                                     a new path
                        }
                        currentPathSet.back ().push_back (myPoint); //
                             pen is down add a new point to the latest
                             path
                }
        }
```

The string 'Jabber' would be encoded as the path 'M24 59l0,16-1,3-1,1-2,1-2,0-2-1-1-1-1-3 0-2M43 66l0,14M43 69l-2-2-2-1-3,0-2,1-2,2-1,3 0,2 1,3 2,2 2,1 3,0 2-1 2-2M51 59l0,21M51 69l2-2 2-1 3,0 2,1 2,2 1,3 0,2-1,3-2,2-2,1-3,0-2-1-2-2M70 59l0,21M70 69l2-2 2-1 3,0 2,1 2,2 1,3 0,2-1,3-2,2-2,1-3,0-2-1-2-2M88 72l12,0 0-2-1-2-1-1-2-1-3,0-2,1-2,2-1,3 0,2 1,3 2,2 2,1 3,0 2-1 2-2M107 66l0,14M107 72l1-3 2-2 2-1 3,0', which is 357 characters long. That is no more than twice the size of a typical groupchat text message.

4.4 Clearing the screen

Some of the protocols mentioned in the introduction, have a clear-screen command. However the benefits of such a command are doubtful. Of course clients can implement such a command locally. A client might even implement finer control such as the possibility of opening new windows that will receive new paths, or showing paths based on whether they were drawn in a selectable timespan. Synchronization of such complex actions between clients is clearly beyond the scope of this protocol. Of course when it is absolutely necessary to clear the screens of both sides in a whiteboard chat, that could be implemented by sending delete-commands for all paths.

5 Security Considerations

There are no security features or concerns related to this proposal.

6 IANA Considerations

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

7 XMPP Registrar Considerations

This document requires registration of the namespace "http://jabber.org/protocol/swb" by the XMPP Registrar [7].

8 Formal Definition

8.1 Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='http://jabber.org/protocol/swb'
    xmlns='http://jabber.org/protocol/swb'
    elementFormDefault='qualified'>

  <xs:element name='x'>
    <xs:complexType>
       <xs:element ref='path' minOccurs='0' maxOccurs='unbounded'/>
       <xs:element ref='move' minOccurs='0' maxOccurs='unbounded'/>
       <xs:element ref='delete' minOccurs='0' maxOccurs='unbounded'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='path'>
    <xs:complexType>
      <xs:attribute name='d' type='xs:string' use='required'/>
      <xs:attribute name='stroke' type='xs:string' use='optional'
          default='#000000'/>
      <xs:attribute name='stroke-width' type='xs:integer' use='
          optional' default='1'/>
      <xs:attribute name='id' type='xs:string' use='optional'/>
```

[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/>.

```
    </xs:complexType>
  </xs:element>

  <xs:element name='move'>
    <xs:complexType>
      <xs:attribute name='id' type='xs:string' use='required'/>
      <xs:attribute name='dx' type='xs:integer' use='required'/>
      <xs:attribute name='dy' type='xs:integer' use='required'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='delete'>
    <xs:complexType>
      <xs:attribute name='id' type='xs:string' use='required'/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

8.2 DTD

```
<?xml version='1.0' encoding='UTF-8'?>
        <!ELEMENT x (path*, move*, delete*) >
        <!ELEMENT path EMPTY >
        <!ATTLIST path d CDATA #REQUIRED
                stroke CDATA #IMPLIED
                stroke-width CDATA #IMPLIED
                id CDATA #IMPLIED >
        <!ELEMENT move EMPTY >
        <!ATTLIST move id CDATA #REQUIRED
                dx CDATA #REQUIRED
                dy CDATA #REQUIRED >
        <!ELEMENT delete EMPTY >
        <!ATTLIST delete id CDATA #REQUIRED >
```

8.3 Grammar of ”d” attribute

The grammar of the ”d” attribute below is a slight simplification of section 8.3.9 in [8].

```
simple-whiteboard-path:
                wsp* moveto-drawto-command-groups? wsp*

        moveto-drawto-command-groups:
                moveto-drawto-command-group
```

[8] Scalable Vector Graphics (SVG) 1.0 Specification, section 8.3.1.: The grammar for path data http://www.w3.org/TR/SVG/paths.html#PathDataBNF

```
        | moveto-drawto-command-group wsp* moveto-drawto-
          command-groups

moveto-drawto-command-group:
        moveto wsp* drawto-commands?

drawto-commands:
        drawto-command
        | drawto-command wsp* drawto-commands

drawto-command:
        lineto

moveto:
        ( "M" | "m" ) wsp* moveto-argument-sequence

moveto-argument-sequence:
        coordinate-pair
        | coordinate-pair comma-wsp? lineto-argument-sequence

lineto:
        ( "L" | "l" ) wsp* lineto-argument-sequence

lineto-argument-sequence:
        coordinate-pair
        | coordinate-pair comma-wsp? lineto-argument-sequence

coordinate-pair:
        coordinate comma-wsp? coordinate

coordinate:
        number

nonnegative-number:
        integer-constant
        | floating-point-constant

number:
        sign? integer-constant
        | sign? floating-point-constant

flag:
        "0" | "1"

comma-wsp:
        (wsp+ comma? wsp*) | (comma wsp*)

comma:
        ","
```

```
        integer-constant:
                digit-sequence

        floating-point-constant:
                fractional-constant exponent?
                | digit-sequence exponent

        fractional-constant:
                digit-sequence? "." digit-sequence
                | digit-sequence "."

        exponent:
                ( "e" | "E" ) sign? digit-sequence

        sign:
                "+" | "-"

        digit-sequence:
                digit
                | digit digit-sequence

        digit:
                "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |
                    "9"

        wsp:
                (#x20 | #x9 | #xD | #xA)
```

9 Conclusion

The present protocol satisfies its basic requirements: it allows for freehand drawing, it is easy to implement, light-weight and it requires no server changes.

10 Acknowledgements

The author would like to thank Alexey Shchepin for helpful comments.