XMPP In-Band Real-Time Text: Real-Time Message Processing Workflow

Sender Software

- Sender's Message Entry (e.g. Text Box)
  - Every time the message updates (text change event)
  - Convert to plain text (if needed)
  - Sender's internal copy of original Unicode real-time message
    - Specific changes to text
    - Generate Action Elements
      - Every transmission interval, generate <msg>
      - Convert to UTF-8
      - Sender transmits to XMPP network

Network

- Recipient receives from XMPP network
  - Parse XMPP Message (XML Processor)
    - Action elements from <msg>
    - Process action elements
      - Asynchronous flow here, if key intervals are supported
        (i.e. flow continues here repeatedly after every key interval action element, via timer or thread)

Recipient Software

- Recipient's internal copy of original Unicode string of sender's real-time message
  - Display Pre-processing of real-time message string
  - Displayed real-time message

- Recommended method is "Monitoring Message Edits" in Section 6.2.1
- This is commonly a GUI element, such as a "Text Box," "Text Field," etc.
- Sometimes text in a control is already automatic-normalized by the operating system.
- Usually in operating system native Unicode format (16-bit or 32-bit)

- Convert graphics (e.g. smileys) to plain Unicode text
- Convert formatting (e.g. XML-IM as per "Supplemental") to plain Unicode text
- Normalization can be done here, but doesn't matter.
- Any other processing that will affect Unicode string, and thus calculations

- Ideally, two internal copies are kept. This copy, as well as the old copy from the previous text change event (to help with "Monitor Message Edits" of Section 6.2.1 of specification)
- If key press method is used, then strict control of message entry is needed to capture all possible changes to text

- Creating action elements includes calculations for positions and lengths (n and p attributes)
- Generation of action element based on comparing previous copy of plain Unicode string with current copy of plain Unicode string, as per Section 6.2.1 of XMPP-RTT specification.
- Includes encoding of key press intervals

- Includes conversion of certain characters to XML entities (i.e. quotes, ampersands)
- After XML is generated, there MUST be no further Unicode processing affecting code points
- Some XML generator also convert to UTF-8 at the time, or the conversion to UTF-8 may be done as a separate step in the software, immediately adjacent to the XML generation

- Recommended transmission interval is every one second.
- We cannot control network performance

- Includes decoding of XML entities, and also conversion of any format of line breaks of inner text into single LINE FEED U+000A. This is required by the XML standard on www.w3.org
- XML processor will usually convert the XML into the programming language's native Unicode format, such as 16-bit or 32-bit Unicode format (UTF16, UCS4)

- Includes processing position and length values (n and p attributes of action elements)
- If recipient client supports key press intervals, this process calls the next process asynchronously of flow from the previous process.

- At this step, this string is identical to the copy of the string on the sender's end, if the XMPP RTT specification is followed, and XML generation/processing is compliant.
- No Unicode code point modification (processing, normalization) should occur between these two storage steps between sender and recipient.

- Optional conversion of text to emoticons, graphics, or other visual decorations
- Includes optional Unicode normalization (if required for this recipient's display)
- Any processing needed for this platform's specific display, that isn't already done by the display step (i.e. GUI control)

- GUI element that displays the processed real-time message
- Many platforms already have GUI controls that automatically handles any Unicode text (normalized or un-normalized), if so, then pre-normalization isn't necessary