Limitations of RFC-5322

- RFC-5322 describes a very limited format.
  - Only a simple text body is allowed.
  - No support for “attachments.”
  - No support for structured text
    - Complex formatted text (like HTML) rendered “as is.”
  - No support for non-ASCII character sets
    - SMTP limitation... messages must be transported by SMTP which only allows ASCII text.
- No support for binary data.
  - SMTP limitation... see above!
MIME

- **Multipurpose Internet Mail Extensions**
  - MIME addresses all of these problems.
  - Defines a way to structure message bodies.
    - To provide for complex formatting (HTML, etc)
    - To provide for attachments
  - Defines a way to encode binary data as ASCII text
    - To provide for non-ASCII character sets.
    - To provide for arbitrary binary data.

- **MIME RFCs**
  - RFC-2045, 2046, 2047, 2048, 2049
    - And a host of others.
MIME Types

- Each message component has a “type.”
  - Top level type categories include
    - Basic types: text, image, audio, video, application
    - Composite types: multipart, message
  - Each category contains specific subtypes.
    - text/plain, text/html, text/enriched
    - image/jpeg, image/gif, image/png
    - application/vnd.ms-excel, application/octet-stream
  - Mail programs can use type information to improve message handling.
    - Images displayed as images, sounds played, etc.
MIME-Version

• MIME enabled messages must say so.
  • New header fields.
    - MIME-Version: 1.0
      Content-Type: text/plain;
      charset="us-ascii"
      Content-Transfer-Encoding: 7bit
  • MIME-Version field is required.
  • Content-Type field specifies the message type.
    - text/plain is the default. ASCII characters are default.
  • Content-Transfer-Encoding specifies how the content is encoded.
    - 7bit (meaning US-ASCII is good enough) is the default.
HTML Mail

- Messages can now be in formatted text.
  - HTML is just one possibility
    - MIME-Version: 1.0
      Content-Type: text/html
      Content-Transfer-Encoding: 7bit

      <html>
      <body>
      <p>Hello! This is email</p>
      </body>
      </html>

- Receiving mail program renders the HTML instead of displaying the raw text.
What About Binary Data?

• There are two primary encodings.
  • Quoted Printable Encoding
    – Encodes most ASCII characters as themselves
    – Non-ASCII characters require three bytes.
      • Inefficient if there are many such characters.
    – Good for “almost” ASCII text.
    – Result still (mostly) readable without decoding.
  • Base 64 Encoding
    – Encodes all byte values into 64 possible ASCII values.
    – Resulting text totally unreadable without decoding.
    – Good for true binary data.
Quoted Printable Encoding

• Characteristics...
  
  • Any byte can be represented as $=XY$ where X and Y are hex digits (00 to FF using uppercase letters).
    – Example: the space can be represented as $=20$
  
  • Most printable ASCII can represent itself (one notable exception is the '=' character, which must be encoded).
  
  • White space can represent itself, but not at the end of a line.
  
  • Line breaks (CF/LF) must be preserved.
Québec

- The string “Québec”...
  - The 'é' has code point U+00E9
  - UTF-8 (all values are hex)
    - Byte values: 51 75 C3 A9 62 65 63
    - In Quoted printable: “Qu=C3=A9bec”
  - UTF-16BE
    - Byte values: 00 51 00 75 00 E9 00 62 00 65 00 63
    - In Quoted printable: “=00Q=00e=00=E9=00b=00e=00c”
- Quoted printable can encode arbitrary binary data.
  - But the size increases by up to 3x.
Soft Line Breaks

- Quoted printable encoding allows long lines to be wrapped.
  - Lines ending with a bare '==' are continued.
    - This is a long = line that has been = wrapped on multiple = lines.
  - Mail programs sometimes put entire paragraphs in one line.
    - So receiving program can wrap them in its window optimally.
    - Yet must prepare message with lines no longer than 78 characters (leaving 2 characters for CR/LF).
Increasing Reliability

- Sending mail can be tricky.
  - Some mail gateways or character set translations have problems with certain characters.

- Recommendation:
  - Use quoted printable encoding to encode ("quote") the characters: `!"#@ [\]^`\`{| |}
  - This reduces the chances of them being corrupted on route to the destination.
  - Note that the above characters can, technically, stand for themselves. This is only a recommendation.
Base64 Encoding

- Starts with raw binary data.
  - Break data into groups of three bytes (24 bits).
  - Divide group into four sections of 6 bits each.

```
+----------------+----------------+----------------+----------------+
<table>
<thead>
<tr>
<th>8 bits</th>
<th>8 bits</th>
<th>8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>6 bits</td>
<td>6 bits</td>
</tr>
<tr>
<td>6 bits</td>
<td>6 bits</td>
<td>6 bits</td>
</tr>
</tbody>
</table>
```
Base64 Alphabet

• 6 bits implies $2^6 = 64$ possibilities.
  • Assign one “safe” character to each of those values.
  • char code_table[ ] =
    “ABCDEFGHIJKLMNOPQRSTUVWXYZ”
    “abcdefghijklmnopqrstuvwxyz”
    “0123456789+/”;

• The three original bytes become four characters from the above alphabet.

• Notice that zero maps to 'A'. Thus a file of zeros becomes “AAAAAAAAAAAA...”
Padding

- What if input not a multiple of three in size?
  - Pad last group of three bytes with zero bits.
  - Use '=' characters as placeholders in output.
    - That way receiver knows those bytes aren't really there.
- Example (padding show underlined):
  - Input bytes: 0x3C 0xA2
  - Binary: 0011,1100 1010,0010 0000,0000
  - Regrouped binary: 001111 001010 0010000000
    - 001111 corresponds to “P”
    - 001010 corresponds to “K”
    - 001000 corresponds to “I”
  - Encoded result: PKI=
Base64 vs Quoted Printable

• Base64...
  • Much more efficient use of space.
    – 3 bytes becomes 4 bytes. Encoded size 133% input size.
    – With quoted printable, encoded size could be as much as 300% input size!
  • Binary data (image data, etc) not readable anyway.

• Quoted printable...
  • Retains readability if most characters are ASCII
Multipart Messages

- **Content-Type:** multipart/mixed

  - Multipart messages contain multiple parts. The “mixed” subtype is used for attachments.
  
  - **Content-Type:** multipart/mixed;
    boundary="fizzle"

  --fizzle
  Content-Type: text/plain
  Content-Transfer-Encoding: 7bit

  The attached file illustrates...
  --fizzle
  Content-Type: image/jpeg
  Content-Transfer-Encoding: base64

  Ay33bkoSk1w/jQLhe8w1clzzA...
  --fizzle--
Multipart Structure

- The body is broken into “parts.”
  - Each part has its own Content-Type and Content-Transfer-Encoding “subheader.”
    - Body of each part separated from the subheader with a blank line.
  - Parts separated by a “boundary line” declared in the main body's Content-Type field.
- Section before the first part is empty.
  - Used for messages seen by non-MIME mail programs: “If you can see this, get a real mail program.”
- Section after last part is empty.
Nested Multipart Messages

• The Content-Type of a part can also be multipart/mixed.

  • Content-Type: multipart/mixed; boundary="fizzle1"

    --fizzle1
    Content-Type: text/plain; charset="utf8"
    Content-Transfer-Encoding: quoted-printable

    blah...
    --fizzle1
    Content-Type: multipart/mixed; boundary="fizzle2"

    --fizzle2
    ... etc ...
    --fizzle2
    ... etc ...
    --fizzle2--
    --fizzle1--
More Nested Fun

• Nesting depth is arbitrary.
  • Nested parts can contain more nested parts.

• Number of parts is arbitrary.
  • A multipart/mixed message can have dozens of parts.
    – All different mime types!
    – Some parts might be nested multipart. Some might be images, HTML, video, etc.

• MIME messages are...
  • A complex tree of parts.
  • Very complex messages can be confusing!
Multipart/Alternative

• Content-Type: multipart/alternative.
  • Structured just like multipart/mixed.
  • Each part intended to be a different representation of the same content.
  • Mail program displays the last part it knows how to handle.
    - Part 1: text/plain (“Let's do lunch!”)
    - Part 2: text/html (“Let's do lunch!” in fancy fonts)
    - Part 3: image/jpeg (Picture of me holding a sign that says “Let's do lunch!”)
    - Part 4: video/mpeg (Video of me doing my “Let's do lunch!” dance)
Handling of Alternatives

• Receiving mail program...
  • IF it can display videos it will show the dance.
  • ELSE IF it can display images it will show the picture.
  • ELSE IF it can display HTML email it will show the fancy fonts.
  • ELSE it shows the plain text.

• Common...
  • Many mail programs send text/plain and text/html alternatives.
    – Although HTML mail is so common now, the text/plain alternative is often dropped.
Content-Type: message/rfc822

- A MIME type for email messages.
  - Used when nesting messages inside of other messages.
    - A multipart/mixed message might contain a part with type message/rfc822.
    - Such parts contain entire email messages... complete with all headers and internal structure
      - Can be multipart messages themselves!
  - Used (sometimes) when forwarding email.
  - Used (sometimes) by mailing lists to create digests.
  - Smart mail programs allow you to reply to the message parts independently. (Pine allows this).
Formal Specification

• MIME is precisely specified.
  • Allows mail programs to reliably perform transformations on email.
    – Break digests into individual messages.
    – Add/remove attachments from messages without corrupting the primary body.
    – Insert/remote/rearrange message alternatives.
    – Perform character set transformations (for example, switching ISO-8859-1 to UTF8)
  • BUT...
    – Digitally signed mail (there's a MIME type for that!) can't be modified without invalidating the signature.
      • Smart mail systems check the MIME type!
Demonstration

Show some MIME messages