Request for Comments

In computer network engineering, a **Request for Comments** (**RFC**) is a memorandum, usually published by the RFC Editor on behalf of the Internet Engineering Task Force (IETF), describing methods, behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems.

Through the Internet Society, engineers and computer scientists may publish discourse in the form of an RFC, either for peer review or simply to convey new concepts, information, or (occasionally) engineering humor. The IETF adopts some of the proposals published as RFCs as Internet standards.

Request For Comments documents were invented by Steve Crocker in 1969 to help record unofficial notes on the development of the ARPANET. They have since become the official record for Internet specifications, protocols, procedures, and events.[1]

The *RFC Editor* assigns each RFC a unique serial number. Once assigned a number and published, an RFC is never rescinded or modified; if the document requires amendments, the authors publish a revised document. Therefore, some RFCs supersede others; the superseded RFCs are said to be *deprecated*, *obsolete*, or *obsoleted by* the superseding RFC. Together, the serialized RFCs compose a continuous historical record of the evolution of Internet standards and practices. For more details about RFCs and the RFC process, see RFC 2026, "The Internet Standards Process, Revision 3".[10]

The RFC production process differs from the standardization process of formal standards organizations such as ISO. Internet technology experts may submit an Internet Draft without support from an external institution. Standards-track RFCs are published with approval from the IETF, and are usually produced by experts participating in working groups, which first publish an Internet Draft. This approach facilitates initial rounds of peer review before documents mature into RFCs.

Sub-series

The RFC series contains three sub-series for IETF RFCs:

BCP
: Best Current Practice; mandatory IETF RFCs not on standards track, see below.

FYI
: For Your Information; informational RFCs promoted by the IETF as specified in RFC 1150 (FYI 1). In 2011, RFC 6360 obsoleted FYI 1 and concluded this sub-series.

STD
: Standard; this used to be the third and highest maturity level of the IETF standards track specified in RFC 2026 (BCP 9). In 2011 RFC 6410 (a new part of BCP 9) reduced the standards track to two maturity levels.

Streams

There are four streams of RFCs: (1) IETF, (2) IRTF, (3) IAB, and (4) *independent submission*. Only the IETF creates BCPs and RFCs on standards track. An *independent submission* is checked by the IESG for conflicts with IETF work; the quality is assessed by an *independent submission editorial board*. In other words, IRTF and *independent* RFCs are supposed to contain relevant info or

experiments for the Internet at large not in conflict with IETF work; compare RFC 4846, RFC 5742, and RFC 5744.

Obtaining RFCs

The official source for RFCs on the World Wide Web is the RFC Editor. Almost any individual published RFC, for example RFC 5000, can be retrieved via the URL: http://www.rfc-editor.org/rfc/rfc5000.txt

Every RFC is submitted as plain ASCII text and is published in that form, but may also be available in other formats. However, as of 2008 the definitive version of any standards-track specification is the ASCII version.

For easy access to the metadata of an RFC, including abstract, keywords, author(s), publication date, errata, status, and especially later updates, the *RFC Editor* site offers a search form with many features. A redirection sets some efficient parameters, example: http://purl.net/net/rfc/5000

The official International Standard Serial Number (ISSN) of the RFC series is 2070-1721.[7]

Status

Not all RFCs are standards.[11] Each RFC is assigned a designation with regard to status within the Internet standardization process. This status is one of the following: *Informational*, *Experimental*, *Best Current Practice* (*BCP*), *Standards Track*, or *Historic* (sic). Standards-track documents are further divided into *Proposed Standard*, *Draft Standard*, and *Internet Standard* documents. The term *Historic* is applied to deprecated standards-track documents or obsolete RFCs that were published before the standards track was established. Only the IETF, represented by the Internet Engineering Steering Group (IESG), can approve standards-track RFCs.

Each RFC is static; if the document is changed, it is submitted again and assigned a new RFC number. If an RFC becomes an Internet Standard (STD), it is assigned an STD number but retains its RFC number; however, when an Internet Standard is updated, its number stays the same and it simply refers to a different RFC or set of RFCs. A given Internet Standard, STD *n*, may be RFCs *x* and *y* at a given time, but later the same standard may be updated to be RFC *z* instead. For example, in 2007 RFC 3700 was an Internet Standard—STD 1—and in May 2008 it was replaced with RFC 5000, so RFC 3700 changed to *Historic*, RFC 5000 became an Internet Standard, and as of May 2008 STD 1 is RFC 5000. When STD 1 is updated again, it will simply refer to a newer RFC that will have completed the standards track, but it will still be STD 1. Best Current Practices work in a similar fashion; BCP *n* refers to a certain RFC or set of RFCs, but which RFC or RFCs may change over time.

The definitive list of Internet Standards is itself an Internet Standard, STD 1: *Internet Official Protocol Standards*.[12]

Status "informational"

An *informational* RFC can be nearly anything from April 1 jokes over proprietary protocols up to widely recognized essential RFCs like Domain Name System Structure and Delegation (RFC 1591). Some informational RFCs formed the FYI sub-series.

Status "experimental"

An *experimental* RFC can be an IETF document or an individual submission to the 'RFC Editor'. A draft is designated experimental if it is unclear the proposal will work as intended or unclear if the proposal will be widely adopted. Experimental RFCs may be promoted to standards track if it becomes popular and works well.[13]

Status "best current practice"

The *best current practice* (BCP) subseries collects administrative documents and other texts which are considered as official rules and not only *informational*, but which do not affect *over the wire data*. The border between standards track and BCP is often unclear. If a document only affects the Internet Standards Process, like BCP 9,[14] or IETF administration, it is clearly a BCP. If it only defines rules and regulations for Internet Assigned Numbers Authority (IANA) registries it is less clear; most of these documents are BCPs, but some are on the standards track.

The BCP series also covers technical recommendations for how to practice Internet standards; for instance the recommendation to use source filtering to make DoS attacks more difficult (RFC 2827: "*Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*") is BCP 38.

Status "historic"

A *historic* RFC is one that has been made obsolete by a newer version, documents a protocol that is not considered interesting in the current Internet, or has been removed from the standards track for other reasons. Some obsolete RFCs are not classified as historic, because the Internet standards process generally does not allow normative references from a standards track RFC to another RFC with lower status. Also, few are interested in working through the required procedural details to get RFCs classified as historic and update all RFCs normatively depending on it.

Status "unknown"

Status *unknown* is used for some very old RFCs, where it is unclear which status the document would get if it were published today. Some of these RFCs would not be published at all today; an early RFC was often just that: a simple request for comments, not intended to specify a protocol, administrative procedure, or anything else for which the RFC series is used today.

How MIME works?

MIME stands for "Multipurpose Internet Mail Extensions". It sound both complicated and meaningless, but MIME extends the original capabilities of internet email in an exciting way.

Email messages have been defined by RFC 822 (and later RFC 2822) since 1982, and they will probably continue to obey this standard for a long time to come.

Nothing But Text, Plain Text

Unfortunately, RFC 822 suffers from a number of shortcomings. Most notably, messages conforming to that standard must not contain anything but plain ASCII text.

In order to send files (like pictures, text processor documents or programs), one has to convert them to plain text first and then send the result of the conversion in the body of an email message. The recipient has to extract the text from the message and convert it to the binary file format again. This is a cumbersome process, and before MIME it all had to be done by hand.

MIME corrects this problem attached to RFC 822, and it makes it possible to use international characters in email messages, too. With the RFC 822 limitation to plain (English) text, this had not been possible before.

The Lack of Structure

In addition to being limited to ASCII characters, RFC 822 does not identify the structure of a message or the format of the data. Since it is clear that you always get one junk of plain text data, this was not necessary when the standard was defined.

MIME, in contrast, lets you send multiple pieces of different data in one message (say, a picture and a Word document), and it tells the recipient's email client what format the data is in so they can make smart choices displaying the message.

When you get a picture, you do no longer have to figure out that it can be viewed with an image viewer. Your email client either displays the image itself or start a program on your computer that can.

Building on and Extending RFC 822

Now how does the MIME magic work? Basically, it employs the cumbersome process of sending arbitrary data in plain text described above. The MIME message standard does not replace the standard laid down in RFC 822 but extends it. MIME messages cannot contain anything but ASCII text either.

This means that all email data must still be encoded in plain text before the message is sent, and it must be decoded to its original format on the receiving end again. The early email users had to do that manually. MIME does it for us comfortably and seamlessly, usually via a smart process called Base64 encoding.

Life as a MIME Email Message

When you compose a message in an email program capable of MIME, the program does roughly the following:

- If the message is in plain ASCII text only, it leaves it alone and only tells the recipient's email client to expect nothing but plain text.
- If the message contains one or more attachments and a body with HTML formatting, each part is looked at and treated separately.

First, the format of the data is determined. This is necessary to tell the recipient's email client what to do with the data, and to ensure proper encoding so nothing is lost during transfer.

Then the data is encoded if it is in a format other than plain ASCII text. In the encoding process, the data is converted to the plain text suitable for RFC 822 messages.

Finally, the encoded data is inserted in the message, and the recipient's email client is informed what kinds of data to expect: Are there attachments? How are they encoded? What format was the original file in?

On the recipient's end, the process is reversed. First, the email client reads the information that was added by the sender's email client: Do I have to look for attachments? How do I decode them? how do I handle the resulting files? Then, each part of the message is extracted and decoded if necessary. Finally, the email client displays the resulting parts to the user. The plain text body is shown in line in the email client together with the image attachment. The program also attached to the message is displayed with an attachment icon, and the user can decide what to do with it. She can save it somewhere on her disk, or start it directly from the email program.

Suggested Reading

- How Base64 Encoding Works
- MIME (Definition)

Base64 encoding makes it possible to send all kinds of data via Internet email.

If the internet is the information highway, then the path for email is a narrow ravine. Only very small carts can pass.

The transport system of email is designed for plain ASCII text only. Trying to send text in other languages or arbitrary files is like getting a truck through the ravine.

How Does the Big Truck go Through the Ravine?

Then how do you send a big truck through a small ravine? You have to take it to pieces on the one end, transport the pieces through the ravine, and rebuild the truck from the pieces on the other end.

The same happens when you send a file attachment via email. In a process known as encoding the binary data is transformed to ASCII text, which can be transported in email without problems. On the recipient's end, the data is decoded and the original file is rebuilt.

One method of encoding arbitrary data as plain ASCII text is Base64. It is one of the techniques employed by the MIME standard to send data other than plain text.

Base64 to the Rescue

Base64 encoding takes three bytes, each consisting of eight bits, and represents them as four printable characters in the ASCII standard. It does that in essentially two steps.

The first step is to convert three bytes to four numbers of six bits. Each character in the ASCII standard consists of seven bits. Base64 only uses 6 bits (corresponding to 2^6 = 64 characters) to ensure encoded data is printable and humanly readable. None of the special characters available in ASCII are used. The 64 characters (hence the name Base64) are 10 digits, 26 lowercase characters, 26 uppercase characters as well as '+' and '/'.

If, for example, the three bytes are 155, 162 and 233, the corresponding (and frightening) bit stream is 100110111010001011101001, which in turn corresponds to the 6-bit values 38, 58, 11 and 41.

These numbers are converted to ASCII characters in the second step using the Base64 encoding table. The 6-bit values of our example translate to the ASCII sequence "m6Lp".

- 155 -> 10011011
- 162 -> 10100010
- 233 -> 11101001

- 100110 -> 38
- 111010 -> 58
- 001011 -> 11
- 101001 -> 41

- 38 -> m
- 58 -> 6
- 11 -> L
- 41 -> p

This two-step process is applied to the whole sequence of bytes that are encoded. To ensure the encoded data can be properly printed and does not exceed any mail server's line length limit, newline characters are inserted to keep line lengths below 76 characters. The newline characters are encoded like all other data.

Solving the Endgame

At the end of the encoding process we might run into a problem. If the size of the original data in bytes is a multiple of three, everything works fine. If it is not, we might end up with one or two 8-bit bytes. For proper encoding, we need exactly three bytes, however.

The solution is to append enough bytes with a value of '0' to create a 3-byte group. Two such values are appended if we have one extra byte of data, one is appended for two extra bytes.

Of course, these artificial trailing '0's cannot be encoded using the encoding table. They must be represented by a 65th character. The Base64 padding character is '='. Naturally, it can only ever appear at the end of encoded data.