Ruby master - Misc #16944

questions about Net::IMAP patches

06/10/2020 03:44 AM - nevans (Nicholas Evans)

Status:	Open
Priority:	Normal
Assignee:	

Description

I work on a product that contains many changes to Net::IMAP, both to the client and especially to the response parser. One part of it is an EventMachine daemon that makes extensive use of Net::IMAP::ResponseParser, but only uses a little bit of the other code from the Net::IMAP client (the client is used in other places though).

I've submitted a few smaller patches already (better CAPABILITY parsing #16626, ID#16610, NAMESPACE#16627), and I have many more that I haven't gotten around to cleaning up for a patch yet (e.g. remembering server-sent CAPABILITIES and even using them in a few places, COMPRESS=DEFLATE, UIDPLUS, SASL-IR, SPECIAL USE, AUTH=XOAUTH2 and AUTH=OAUTHBEARER, OBJECTID, UTF8=ACCEPT, ENABLE, and many many RFC3501 parser bugs).

But before I prepare some of the bigger patches, I should ask:

- I would find it easier if net/imap.rb were split into several files. Would a patch like that be welcome and accepted?
  - especially: net/imap/response_parser.rb for Net::IMAP::ResponseParser.
  - But also: net/imap/response_data.rb for all of the response data Structs (e.g. TaggedResponse, FetchData, MailboxList, etc),
  - and also: net/imap/request_data.rb (e.g. all of the #send_data/#validate classes, like Atom, Literal, QuotedString, etc).
- I would really like to split Net::IMAP::ResponseParser into three parts: the parser, the lexer, and a "builder" (so the data structures built by the parser can be more easily customized). Would a patch that does just this (and nothing else) this be accepted?
- I have made many adhoc changes to the response parser over the years as I've worked around various server bugs or bugs in the parser. But the biggest issue seems to be that the parser is still largely based on an obsolete IMAP dialect. It's clear from looking at the comments and code around astrings (just as one example) that much of it pre-dates RFC3501 and was written to the older RFC2060 standards. Although I can continue creating extracting and cleaning up smaller patches to update smaller pieces, would a much larger cleanup project to more closely match RFC3501 be accepted?
- And lastly, what do you think about rewriting the parser/lexer using racc and/or ragel? Unlike the above, I haven't even started on this project yet. And I wouldn't want to even start on this project until after the above has been taken care of, and the test suite has been added to. However, CPU is the limiting resource for my IMAP daemons, and I've wanted to rewrite or replace our parser for years now. Even using EventMachine on some very powerful servers, I usually can't hold more than ~500 mostly idle IMAP connections per OS process before CPU is fully maxed (*far* fewer if they are very active at once). For our newest project, my team has been using golang instead of ruby, even though the go IMAP library has fewer features and is more difficult to work with. That makes me sad. :(

History

#1 - 06/10/2020 06:42 AM - shevegen (Robert A. Heiler)

I believe your question is for the ruby core team, so probably it is best to let nobu or shyouhei or anyone else from the core team who would like to comment.

Personally I think what you described above, provided that it is what is then changed (have to match the proposals to the actual changes :)) is IMO fine. Some of the code/the files is probably very old. Perhaps the core team may see this as an opportunity if someone takes a closer look at the code.

The last point seems a bit more challenging. For that having benchmarks available may help - see discussions about jemalloc on the tracker here for example and proposals by byroot and others about other parts of making parts of ruby faster/more efficient.

#2 - 06/16/2020 02:28 PM - nevans (Nicholas Evans)

Thanks @shevegen. Is there a better way to send my questions to nobu (Nobuyoshi Nakada) or shyouhei (Shyouhei Urabe), other than posting questions here in this issue tracker, and perhaps tagging them? :)