JMAP @ IETF 99

18 July 2017, 13:30, Prauge

Chairs:
Bron Gondwana – brong@fastmailteam.com
Barry Leiba – barryleiba@computer.org

Authors:
Neil Jenkins – neilj@fastmailteam.com
Dave Crocker – dcrocker@bbiw.net
Neil Jhaveri – neil@neiljhaveri.com

Hello and Welcome

- Introductions
- Blue sheets
- Note Well
- Scribe and Jabber

Note Well

Any submission to the IETF intended by the Contributor for publication as all or part of an IETF Internet-Draft or RFC and any statement made within the context of an IETF activity is considered an "IETF Contribution". Such statements include oral statements in IETF sessions, as well as written and electronic communications made at any time or place, which are addressed to:

- **The IETF plenary session**
- **The IESG, or any member thereof on behalf of the IESG**
- **Any IETF mailing list, including the IETF list itself, any working group or design team list, or any other list functioning under IETF auspices**
- **Any IETF working group or portion thereof**
- **Any Birds of a Feather (BOF) session**
- **The IAB or any member thereof on behalf of the IAB**
- **The RFC Editor or the Internet-Drafts function**

All IETF Contributions are subject to the rules of RFC 5378 and RFC 8179.

Statements made outside of an IETF session, mailing list or other function, that are clearly not intended to be input to an IETF activity, group or function, are not IETF Contributions in the context of this notice. Please consult RFC 5378 and RFC 8179 for details.

A participant in any IETF activity is deemed to accept all IETF rules of process, as documented in Best Current Practices RFCs and IESG Statements.

A participant in any IETF activity acknowledges that written, audio and video records of meetings may be made and may be available to the public.

Agenda for today

- Overview of goals: 5 min
- JMAP Mail: 30 min
  - keywords 5 min
  - submission 10 min
  - partial body fetch and json message structure 15 min
- JMAP Core: 50 min
  - extension mechanisms 15 min
  - authentication 10 min
  - push mechanism 10 min
  - transport (json over http) 15 min
- Other business / github tickets: 30 min

Why JMAP?

- IMAP is not well suited to constrained network environments and mobile.
- It’s too hard to write an MUA with current standards, which has led to a stagnation in good email clients.
- Many proprietary JSON over HTTP protocols for email have been appearing.

Charter and Scope

- Specify a single, efficient protocol for email clients to communicate with one server endpoint.
- Address mobile / poorly connected / constrained environment considerations.
- Maintain compatibility with IMAP data models to allow a server to support both protocols.
- Out of scope: server to server communication or new end-to-end encryption.

Goals for JMAP

- Efficient use of network
  - Minimise round trips
  - Minimise bandwidth usage
  - Control over batch sizes
  - Out of band push to avoid polling or long-running connections
- Ease of implementation
  - Developer friendliness (existing platform support)
  - Extensible data model to support additional datatypes (e.g. calendars and contacts)
- Data model compatibility with IMAP
  - Can implement both in one server and provide simultaneous access via both protocols

JMAP Mail: Keywords

- Seems concluded
- https://github.com/jmapio/jmap/pull/61
- https://github.com/jmapio/jmap/issues/73
- Extendable via IMAP keywords registry or abitrary user-defined keywords.
- Uses $Seen rather than \Seen

JMAP Mail: Keywords

- "keywords": {
  "$Seen": true,
  "$Answered": true,
  "custom": true
  }
- Replaces isUnread, isDraft, etc.

JMAP Mail: Submission

- Changed to use a more JMAP-ish MessageSubmission object
- https://github.com/jmapio/jmap/pull/99
- Ongoing discussion on the mailing list, but we think it’s close to consensus.

```
{
    "id": "1234-583-134",
    "identityId": "31234-123",
    "messageId": "13410-13123-4",
    "threadId": "40123920-12-3-11",
    "envelope": {
        "mailFrom": "james@example.com",
        "rcptTo": ["jane@example.com", "bill@example.com"]
    },
    "maybeCanUnsend": true,
    "deliveryStatus": {
        "jane@example.com": {
            "code": 250,
            "text": "2.0.0 OK: queued as 44g23f2",
            "isFinal": true
        },
        "bill@example.com": {
            "code": 452,
            "text": "4.2.2 The email account tried to reach is over quota",
            "isFinal": false
        }
    }
}
```

JMAP Mail: Message object & Partial Body Fetch

- Interesting case – message with megabytes of plain text body.
- May not want to download the entire body text.
- Message content is immutable, so OK to provide multiple fetching methods.
- Experience with JCAL et al, raw mapping structured data to JSON gives little benefit.
- No proposal yet.

JMAP Core: extension mechanisms

- How do you specify which extensions are used?
- Proposal from mailing list: top level object with space for additional metadata about request rather than top level being a list of method calls.
- “using”: list of named extensions – e.g. jmapcalendar, fastmail.com/accountmanagement.
- Vendor extensions namespaced.
- No x- prefixes, extensions need to be compatible.
- Unless opt-in, must conform to base spec only.
- “Using jmapmail”.

JMAP Core: authentication

- Have been advised that defining a general Authentication mechanism inside JMAP-core is a bad idea and will be rejected during last-call.
- Would love to find something already existing that isn’t just raw username/password and doesn’t require every client implementation registering with every service separately.
- Plan: remove authentication from core spec, leaving just endpoint discovery and assuming already authenticated somehow, propose separate spec document for auth.

JMAP Core: authentication

- Needs to meet the following criteria:
  - Is session based (creates an authentication token rather than allowing username/password directly).
  - Does not require the client writer to have manually registered with the JMAP server.
  - Allows common 2FA factors to be used (e.g. TOTP, U2F).
  - Be extensible with further authentication mechanisms in the future.
  - Doesn't require a browser in order to preform authentication.

JMAP Core: push mechanism

- Back to the goals – need to allow out-of-band push so we're not holding a connection open on mobile platforms (particularly APNS / GCM)
- Needs to transfer state string to avoid extra fetch after every change
- Want to add: isDelivery (aka "you've got mail") flag so clients can choose when to notify.
- Also needs to work on web/desktop, so support long polling or in-band push.
- Needs to be generic enough to support new push standards as they appear.
- Open question: does the push data need to be encrypted?

JMAP Core: transport

- Area of much debate on mailing list.
- HTTP as a tunnel vs HTTP resources for every object in JMAP and shenanigans to reduce round trips using parallel connections and HTTP/2 to keep latency down.
- https://mnot.github.io/I-D/bcp56bis/
- Email is structured, so relationships are consistent.

JMAP Core: transport

- Why not just tunnel?
  - Caching for server scalability, latency and bandwidth reduction, and reliability;
  - Authentication and access control;
  - Automatic redirection;
  - Partial content to selectively request part of a response;
  - Natural support for extensions and versioning through protocol extension; and
  - The ability to interact with the application easily using a Web browser.

JMAP Core: transport

- Why array-of-JMAP-methods
  - Backreferences to create and use in same call
  - Batch updates in single call (rename a=>b, b=>a)
  - Server doesn’t need to hold any state between requests
  - Simple to implement clients on any platform, library support is easy, debugging is clear compared to concurrent requests
  - Atomic fetch of multiple resources at same server state
  - Easier for server to implement rate limiting
  - Fewer round trips than even most complex HTTP/2-feature-using suggestion.

Github tickets

- https://github.com/jmapio/jmap/issues
- https://github.com/jmapio/jmap/issues/78
  - changed vs created in getFooUpdates
  - some servers may not know when something was created, only if it was updated
  - client can check if it already has it, otherwise needs to fetch either way

Github tickets

- https://github.com/jmapio/jmap/issues/69
  - Disallow destroying mailboxes with messages in them.
  - Upside: deleting more data requires more effort, this is generally good for data safety
  - Downside: need to request the full list of messages in the mailbox and explicitly remove said mailbox from each before it’s possible to remove the mailbox.

mailboxIds property

- Now that keywords exists and is a map (aka unorded set), should align mailboxIds as well.
- Will make it easier to update with a patch syntax and remove a mailboxId from a message without affecting the remaining mailboxes on that message.
- Philosophical issue: message with no mailboxIds.

Any other business?

- Other github tickets of interest?
- Hackathon in Singapore
- JMAP dinner tonight
- Testing
- Calendar/Contact – future extensions

Useful links

- https://github.com/jmapio/jmap (spec & issues)
- http://jmap.io/ (background & implementation)
- https://proxy.jmap.io/ (proxy to IMAP/*DAV)
- https://datatracker.ietf.org/wg/jmap/documents/