IETF123 - JMAP Filenode

Bron Gondwana <brong@fastmailteam.com>

A basic filesystem

- A tree of nodes, with parentId linkage
- Goal is to match WebDAV or SFTP in terms of features
- Some research with Fastmail's server suggested that modifiedTime was the most common custom field that many clients use
- Mirrors Fastmail's filestorage API but generalised a bit.

ACLs

- Do we need more than POSIX style permissions? mayWrite, mayRead, mayAdmin.
  - Also maybe mayExecute?
  - mayCreateChild from JMAP Mail as well?

- Are ACLs recursive?
  - I propose: they copy parent by default if unspecified.

Node Types

- Are “Directory” and “File” enough?
- Maybe Symlinks as well.
- Anything else?

Changed

- Do we need to handle “Changed” the same way that Mailboxes in JMAP Mail work? Or more like inodes in POSIX?
  - Is a Folder/Directory Node changed if it’s children change?
  - How about is a node changed if one of its ancestors changes name? (so the full path is invalidated)

In Mail, a mailbox change is rare enough that all child mailboxes needing a refresh makes sense, but in Files maybe that’s obnoxious!

Blob Diffs

- When creating a synced remote filesystem, sometimes changes to files are much smaller than the entire file. If you already had the old copy, you could calculate a diff!
  - rdiff support for getting changes?
  - just request a binary diff between two files? With local caching you might still have the old file as well, so you can calculate the diff locally!
  - VCDIFF (RFC 3284) binary diff?

- With this, we should be able to create a reasonably efficient remote filesystem.

Client Implementation

- I'd love to have open source implementations of:
  - Microsoft's Cloud Files API client for Windows
  - Apple's File Provider API for Mac and IOS
  - Google File Provider API for Android
  - A FUSE filesystem for Linux

- JMAP is quite amenable to proxying, so a JMAP server could present multiple different source servers as separate accounts, and allow copying between them.
  - Do we need to look at OpenCloudMesh as presented at DISPATCH?

Generic Metadata

- Do we need it? Pros are “clients can store stuff which servers don’t know about, so they don’t need local storage or hacks of putting their config in “files” in the filesystem which is .DS_Store ugly.

- Con: everyone does their own thing and it doesn’t interoperate. The “standard JMAP way” is to define an extension which the server supports – but this means that servers always need to move first!

Next steps

- I want to get a couple of different servers and a couple of different clients including cloud APIs for popular operating systems so we can get some real world testing to be sure our API is complete and performant enough.