# Versioning for HTTP Resources

draft-toomim-httpbis-versions-00

Michael Toomim

Braid.org

Invisible College

IETF 120, Vancouver

# Review from IETF 118

Braid-HTTP: Four independent extensions to HTTP. From **state transfer** to **state synchronization**.
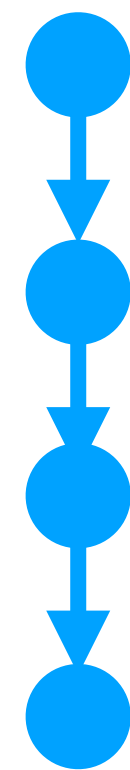
1. Versioning ⟵ Today!
2. Subscriptions
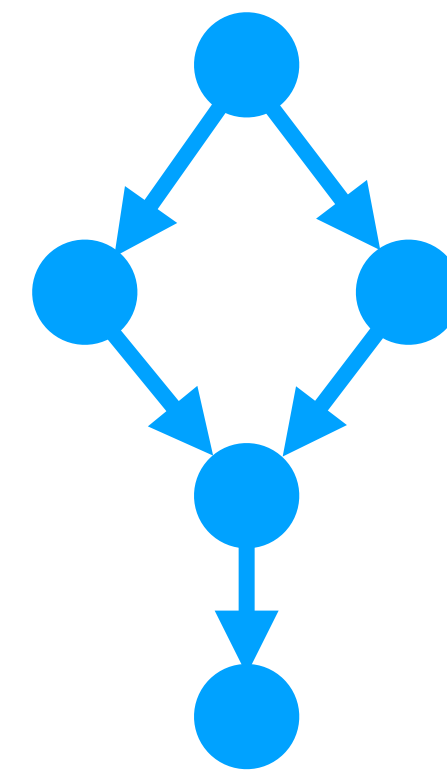3. Patches
4. Merge-Types

# **Versioning of HTTP Resources** *not APIs!*

HTTP resources change over time. Each change to a resource creates a new **version** of its state.

Version history can be **linear**

Distributed system history forms a **DAG**

# Resource Versioning in HTTP today

## 1. Last-Modified

`Last-Modified:` **`Sat, 6 Jul 2024 07:28:00 GMT`**

## 2. ETag

`ETag:` **`W/"2u34fa7yorz0"`**

## 3. Additional URLs     e.g. in WebDAV, Memento, Link Relations

`https://unpkg.com/braid-text@`**`0.0.18`**`/index.js`

# Limits of Today's Versioning

| | Last-Modified | ETag | Additional URLs |
|---|---|---|---|
| Store & Access History | | | ✅ |
| Marks Time vs. Contents | ✅ | | ✅ |
| Branching (Distributed) Time | | | ~ |
| Independent of URL Allocation | ✅ | ✅ | |
| Customizable Time Formats | | | ~ |

# Abilities of Generalized Versioning

- Incremental RSS Updates

- Git repo as website

- CDN with historical archive

- Apply PATCH at old versions

  - Rebase for other clients

- Send PUT/PATCH/POST edits while offline

- Collaborative Editing

# Design Goals

- Version marks **point in time**

- Time can **branch** and **merge** (distributed)

- History of versions can be **stored** and **accessed**

- **Extensible** to new time **formats** (optimizable!)

- Additional URLs not required

**Discussion:** Is the WG interested in this problem?

# Resource Versioning Proposal

- **Version:** and **Parents:** headers

  - Version = set of Version IDs

- **Version ID** = unique string identifying an edit

- **Version-Type:** defines format of string (optimizations!)

- Extend **GET/PUT/POST/PATCH** with version semantics

# Version: and Parents: headers

```
PUT /foo
Version: "dkn7ov2vwg"
Parents: "ajtva12kid", "cmdpvkpll2"

"Hello world!"
```

```
GET /foo
Version: "dkn7ov2vwg"
```

```
HEAD /foo
```

```
HTTP/1.1 200 OK
Version: "dkn7ov2vwg"
```

# Varieties of **GET**

```
GET /foo
```

```
GET /foo
Version: "dkn7ov2vwg"
```

```
GET /foo
Parents: "ajtva12kid"
```

```
GET /foo
Version: "dkn7ov2vwg"
Parents: "ajtva12kid"
```

# Varieties of **PUT/POST/PATCH**

```
PUT /foo
```

```
PUT /foo
Version: "dkn7ov2vwg"
```

```
PUT /foo
Parents: "ajtva12kid"
```

```
PUT /foo
Version: "dkn7ov2vwg"
Parents: "ajtva12kid"
```

# Multiresponse & **Current-Version**

```
GET /foo
Version: "3"
Parents: "1a", "1b"
```

```
HTTP/1.1 104 Multiresponse
Current-Version: "3"

HTTP/1.1 200 OK
Version: "2"
Parents: "1a", "1b"

"Hello!"


HTTP/1.1 200 OK
Version: "3"
Parents: "2"

"Hello World!"
```

# Advanced features via **Version-Type**

```
Version-Type: vector-clock
Version: "{mike: 5, seph: 10, greg: 3}"
```

```
Version-Type: rle
Version: "mike-573"
Parents: "mike-400"
```

Enables **1000x** history compression!

```
Version-Type: git
Version: "1d5f89f8843b33b91d62bf95877e46b23fd86741"
Parents: "1cf6ab4ed836d4d7308ac93edbc6fd18a69ef88f"
```

# Example: Resumeable Uploads

```
PUT /something
Current-Version: "900"
Version-Type: bytestream
Content-Length: 900

<binary data of length 900>
```

For a bytestream:
**Version** = **length-of-stream**!

```
HEAD /something
Parents: "0"
```

How much have you received?

```
200 OK
Parents: "0"
Version: "500"
```

I have up to 500!

# Example: Git

```
GET /repo/readme.md
```

```
HTTP/1.1 200 OK
Content-Type: text/markdown
Version-Type: git
Version: "9531a9702af0d90dd489050ed8e25f87912a9252"
Parents: "3a4c361f8e0349fe4b25c1ff46ebec1cec66e60f"

...
```

# Implemented in Production

# Generalized Versioning of HTTP Resources

**Discussion:**

- Q1: Interest in the problem?

- Q2: Is a solution feasible?

- Q3: Adopt?