HTTP Version Translation of the Capsule Protocol

Ben Schwartz, Meta
Kazuho Oku, Fastly

HTTPBIS, IETF 123

HTTP/1.1 Upgrade ⇔ Extended CONNECT

```
GET /foo HTTP/1.1
Connection: Upgrade
Upgrade: connect-ip
...
```

⇔

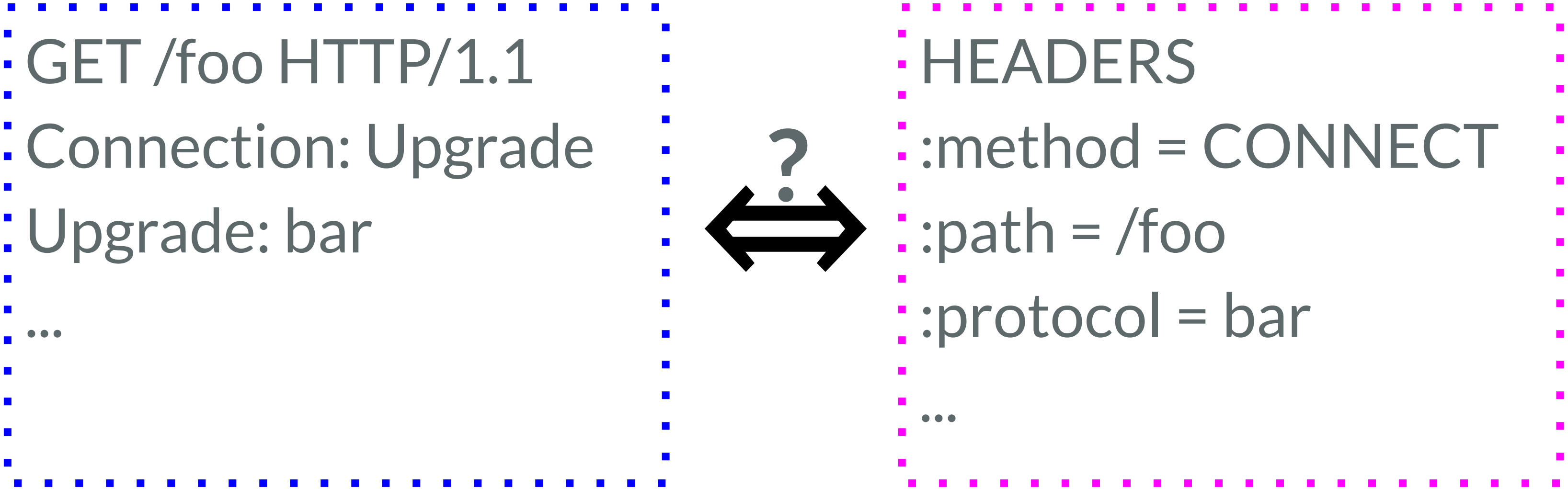
```
HEADERS
:method = CONNECT
:path = /foo
:protocol = connect-ip
...
```

HTTP/1.1 Upgrade vs. Extended CONNECT

GET /foo HTTP/1.1
Connection: Upgrade
Upgrade: **bar**
...

HEADERS
:method = CONNECT
:path = /foo
:protocol = **bar**
...

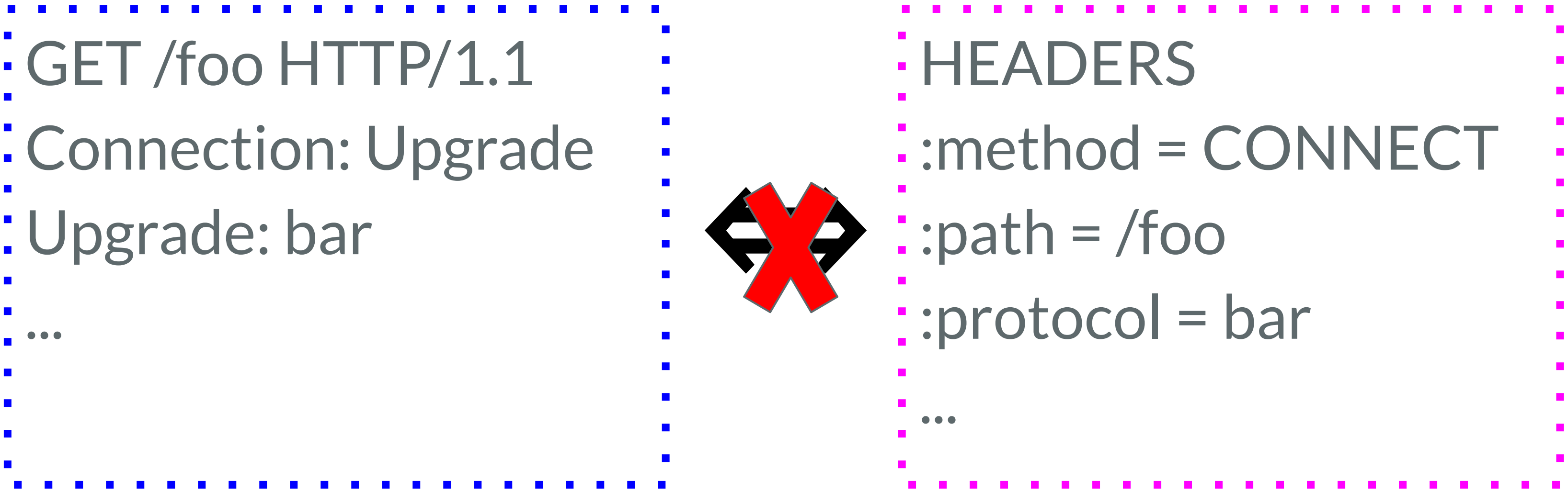
Can an intermediary convert them?



```
GET /foo HTTP/1.1
Connection: Upgrade
Upgrade: bar
...
```

```
HEADERS
:method = CONNECT
:path = /foo
:protocol = bar
...
```

Can an intermediary convert them? NO



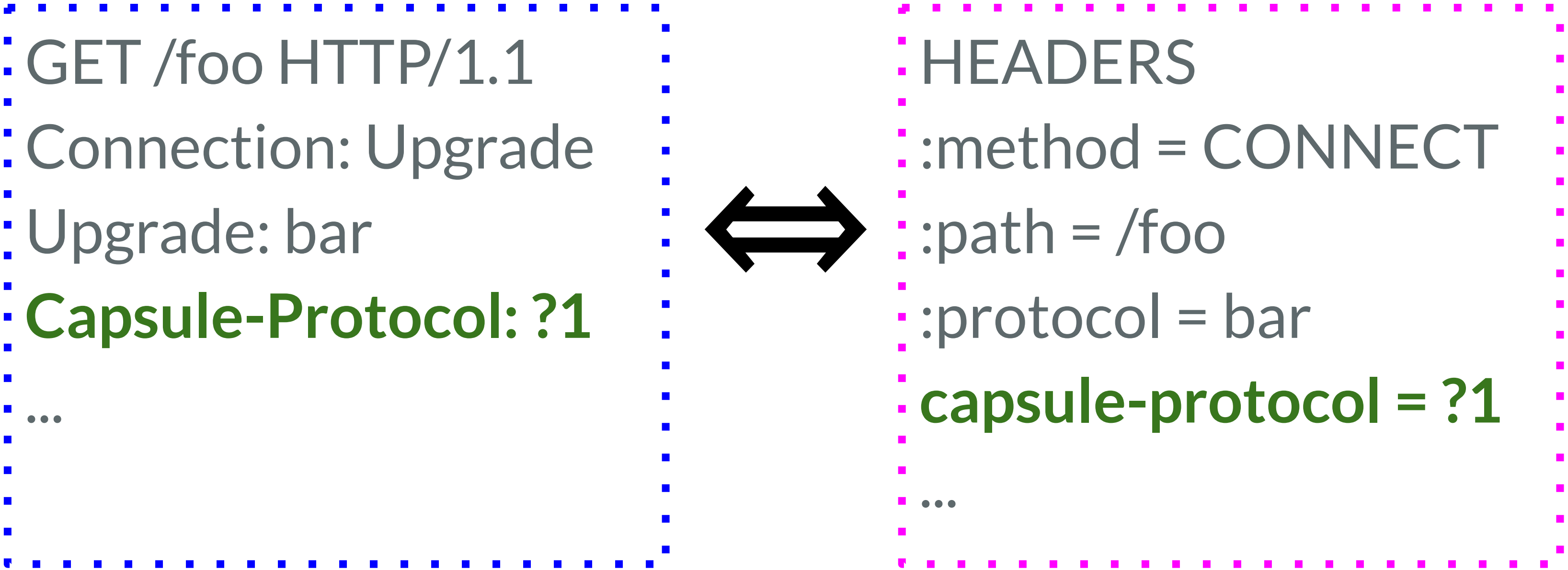
```
GET /foo HTTP/1.1
Connection: Upgrade
Upgrade: bar
...
```

```
HEADERS
:method = CONNECT
:path = /foo
:protocol = bar
...
```

Why not?

- Some existing Upgrade Tokens (HTTP/*.*, TLS/*.*, h2c) preserve the method.
- We haven’t promised any restrictions on how future Upgrade Tokens might behave.

What if the request uses **the Capsule Protocol**?

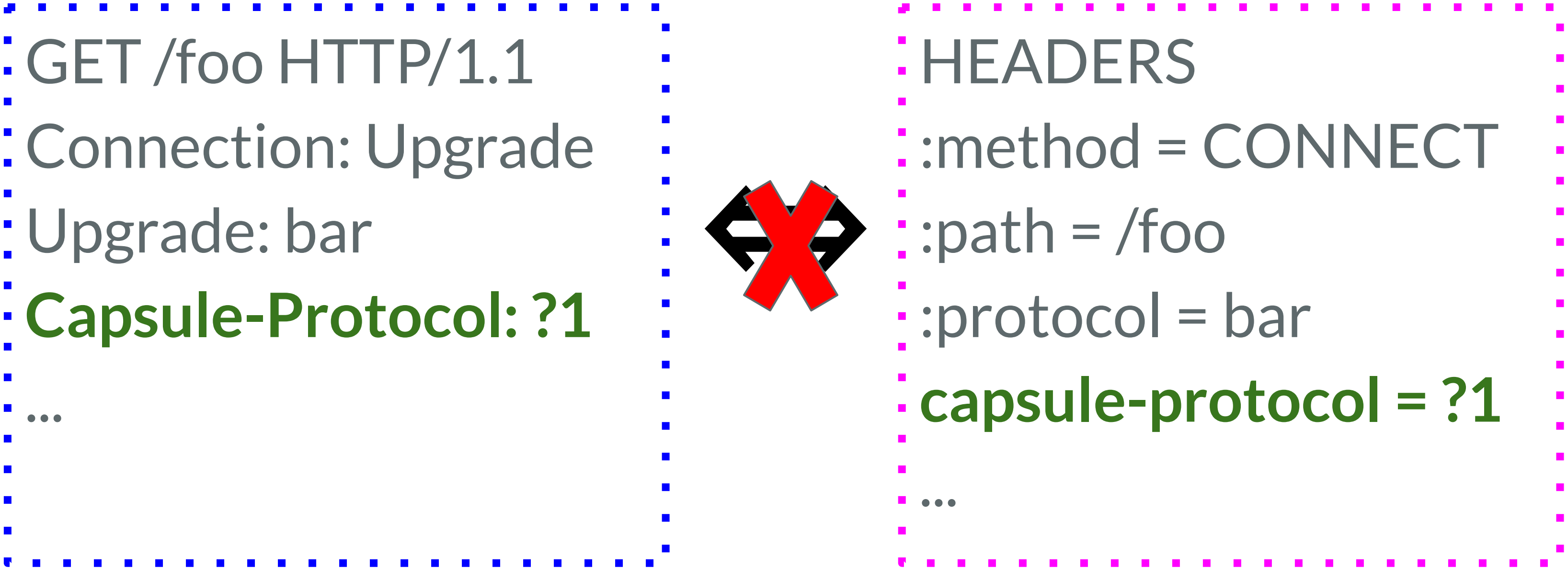


```
GET /foo HTTP/1.1
Connection: Upgrade
Upgrade: bar
Capsule-Protocol: ?1
...
```

⇔

```
HEADERS
:method = CONNECT
:path = /foo
:protocol = bar
capsule-protocol = ?1
...
```

Still No



GET /foo HTTP/1.1
Connection: Upgrade
Upgrade: bar
Capsule-Protocol: ?1
...

HEADERS
:method = CONNECT
:path = /foo
:protocol = bar
capsule-protocol = ?1
...

Why not?

- ~~Some existing Upgrade Tokens (HTTP/*.*, TLS/*.*, h2c) preserve the method.~~
- We haven’t promised any restrictions on how future Upgrade Tokens might behave.
- **RFC 9297 hasn’t promised any restrictions on what future capsuleish protocols might do**
    - **(except that DATAGRAM capsules can always be converted to HTTP/3 DATAGRAM frames)**

This is annoying

- (Multi-version) intermediaries need to maintain a list of all Capsule Protocol Upgrade Tokens (CPUTs).
- New CPUTs won’t work correctly until intermediaries are updated.
  - Even when intermediaries aren’t being asked to do anything interesting.
- Increased uncertainty for designers of new CPUTs.
- Strong temptation to run everything over WebSocket or something.

Can we fix it?

YES!

Proposal

- Declare that Capsule-Protocol requests can be translated freely between HTTP versions in the obvious way, even when the CPUT is unrecognized.
- Accept various implications for our future selves:
    - All future CPUTs are stuck with the GET/CONNECT method.
    - The "Capsule-Protocol" response header has no effect.
    - CPUTs cannot change intermediaries' treatment of capsule types.
    - CPUTs cannot change the meaning of an HTTP extension.
    - Capsule Protocol endpoints must be semantically version-independent.
    - CPUTs that really care about version translation will have to change “SHOULD send” to “MUST”.
    - A "Capsule-Protocol: ?0" request header cannot be used to disable HTTP version translation.

Questions and Quandaries

- What does this mean for WebTransport?
    - We would need to include an exemption for “webtransport” in the definition of SETTINGS_WT_MAX_SESSIONS.
- Alternate spellings
    - Capsule-Protocol: ?1; convertible
    - Version-Independent-Upgrade: ?1