WebPush

IETF 94
Subscription Sets
One GET to GET them all

https://github.com/webpush-wg/webpush-protocol/issues/41
https://github.com/webpush-wg/webpush-protocol/issues/43
https://github.com/webpush-wg/webpush-protocol/issues/46
Changes to Requesting Subscriptions

User Agent

I can has Subscription?
urn:ietf:params:push = Push1
urn:ietf:params:push:set = Set1

Push Service

Provide Subscription (Push1)

I can has Subscription?
urn:ietf:params:push = Push2
urn:ietf:params:push:set = Set1

Application Server(s)

Provide Subscription (Push2)
Changes to Receiving Messages

User Agent

Push Service

Application Server(s)

Push Message to Push1

Push Message to Push2

Monitor (Set1)

Push Message
:urn:ietf:params:push = Push1

Push Message
:urn:ietf:params:push = Push2
That doesn’t correlate

Draft pull request available for review:

https://github.com/webpush-wg/webpush-protocol/pull/53

but how does the Push Service correlate two subscriptions?

https://github.com/webpush-wg/webpush-protocol/issues/44
Push Message Replacement

https://github.com/webpush-wg/webpush-protocol/pull/12
sent message, changed mind

Recall doesn’t help if it has been delivered, but if not
In case of mistake:
  DELETE <push message URI>
In case of new information:
  PUT <push message URI>
    ... new message content ...
Enter the race condition
race condition - ack happens first

User agent deletes message

... shortly after, an update arrives

Recommendation: try to deliver the “new” message

Possible drawback: the “new” message will have the same URL as the “old” message
race condition - update happens first

A push message is updated

... immediately afterwards, the user agent deletes the previous message

... updated message is lost

Recommendation: recommend using conditional DELETE for ack

... https://github.com/webpush-wg/webpush-protocol/pull/54
POST to update

Acts like PUT when the message is still present
Acts like POST to push subscription endpoint when not present

Probably better on balance:
  + Solves the race at the server
  + No need for etag
  - Not idempotent any more
  - Application server needs to look for status code
  - User agent won’t see collapse (but they probably wouldn’t have anyway)
Negative Acknowledgements

https://github.com/webpush-wg/webpush-protocol/pull/12
512 (Expired Resource)

Proposed new HTTP status code for negative acknowledgements:
https://github.com/webpush-wg/webpush-protocol/pull/50

but:
Status codes are generic; they are potentially applicable to any resource, not just one particular media type, kind of resource, or application of HTTP. As such, it is preferred that new status codes be registered in a document that isn't specific to a single application. [RFC7231 -8.2.2. Considerations for New Status Codes]

Alternative (s):
• 404 (Success) and 410 (Failure)
• 4xx + Acknowledgement-Data header field
Acknowledgement-Data

https://github.com/webpush-wg/webpush-protocol/issues/42
https://github.com/webpush-wg/webpush-protocol/pull/57
Before

User Agent

Push Service

Application Server

Monitor Push Subscription

Push Message

Push Receipt

Acknowledge Push Message

Push Message

Push-Receipt:

Monitor Receipt Subscription

Push Receipt
After

User Agent

Push Service

Application Server

Monitor Push Subscription

Push Message

Push-Receipt:

Acknowledgement-Data:

Acknowledge Push Message

Push Message

Push-Receipt:

Monitor Receipt Subscription

Push Receipt

Acknowledgement-Data:
What’s the room temperature?

- Message Priorities
  https://github.com/webpush-wg/webpush-protocol/issues/28

- Port Numbers
  https://github.com/webpush-wg/webpush-protocol/issues/37

- Acknowledging Acknowledgements
  https://github.com/webpush-wg/webpush-protocol/issues/39

- Separate TTL(s) for Messages and Acknowledgements
  https://github.com/webpush-wg/webpush-protocol/issues/40