WebPush @ IETF 95

Brian Raymor

State of Play

The Past – IETF 94 Minutes

- webpush-03 milestone
  - Added explicit correlation to subscription sets
  - Added push message updates (message collapsing)
- webpush-04 milestone
  - Required Application Servers to include TTL header
  - Added push message urgency
- Closed Acknowledgement Data issue and pull request

The Present – IETF 95

- webpush-05 milestone
  - Issue 44 - Sender and Client Authentication

    Related agenda item: Voluntary Application Server Identification

  - Issue 75 - Improve Security Considerations section
    - Informative reference to content encryption
    - Informative reference to vapid authentication (pending adoption)

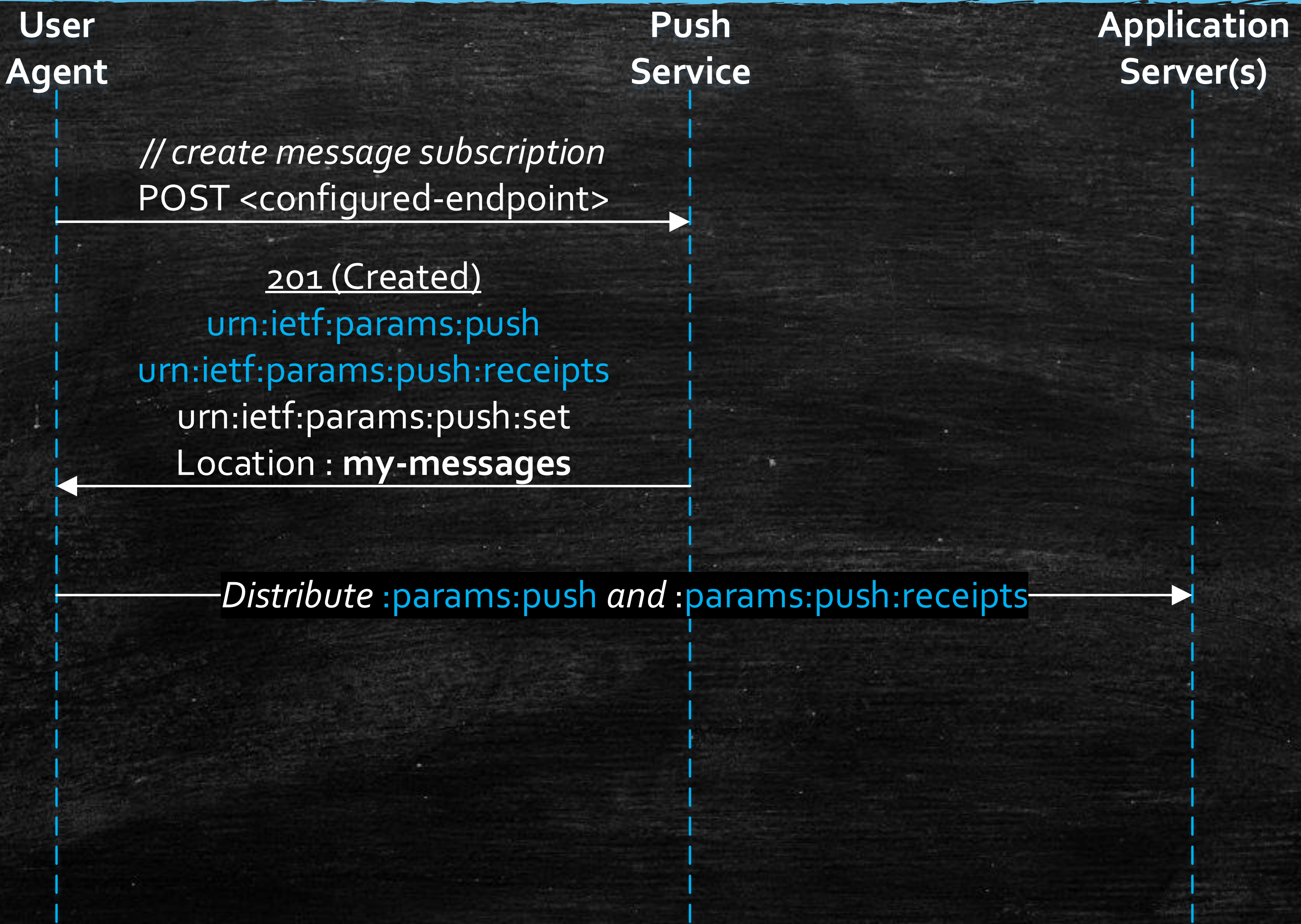
    Related agenda item : Content Encoding Status
    Related agenda item : Voluntary Application Server Identification

  - Issue 81 - Simplifying Acknowledgements

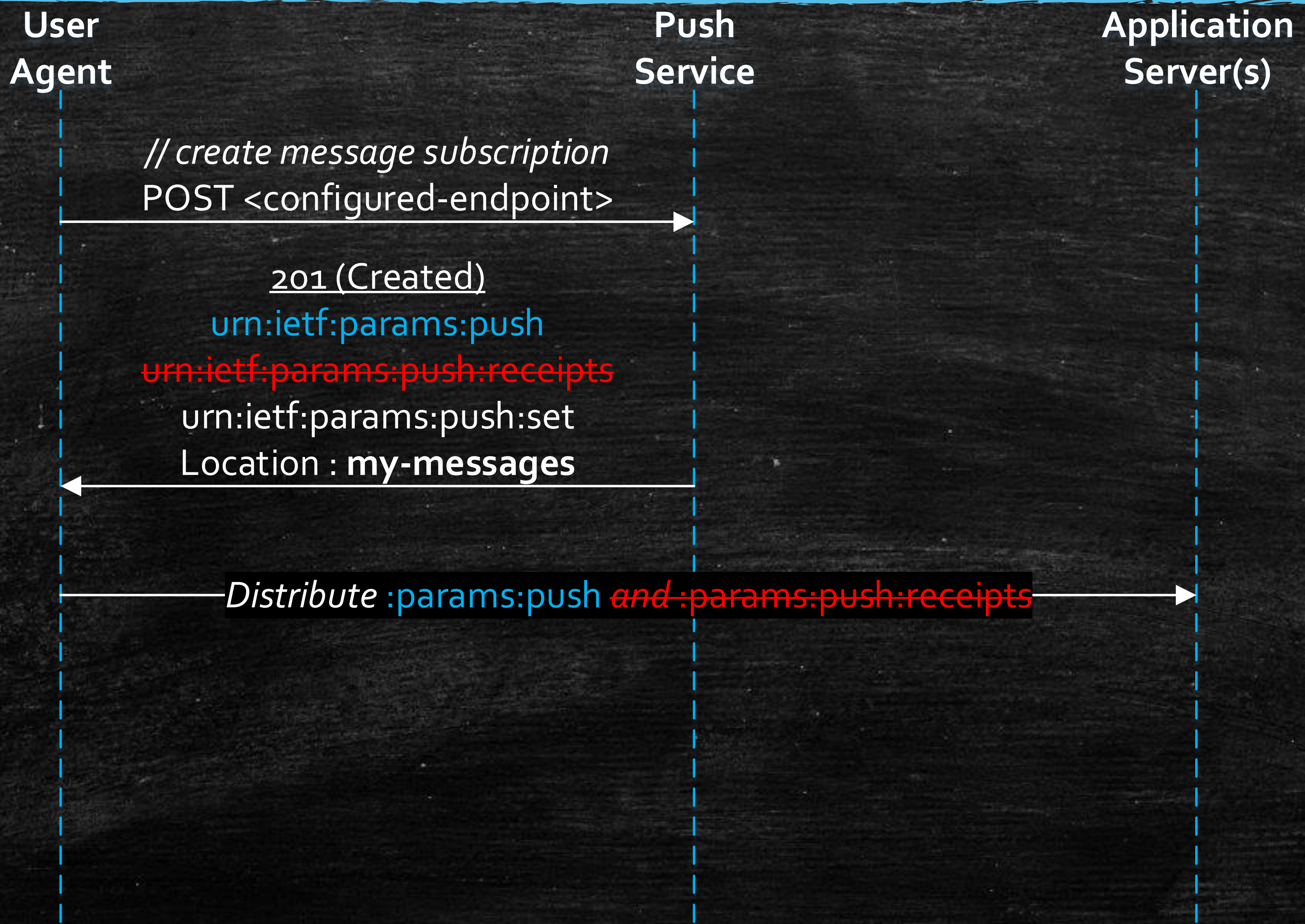
Simplifying Acknowledgements

Eliminates the :params:push:receipts link relation and its machinery

Subscribing to Messages (Before)


User Agent
Push Service
Application Server(s)
// create message subscription
POST <configured-endpoint>
201 (Created)
urn:ietf:params:push
urn:ietf:params:push:receipts
urn:ietf:params:push:set
Location : **my-messages**
Distribute :params:push *and* :params:push:receipts

Subscribing to Messages (After)

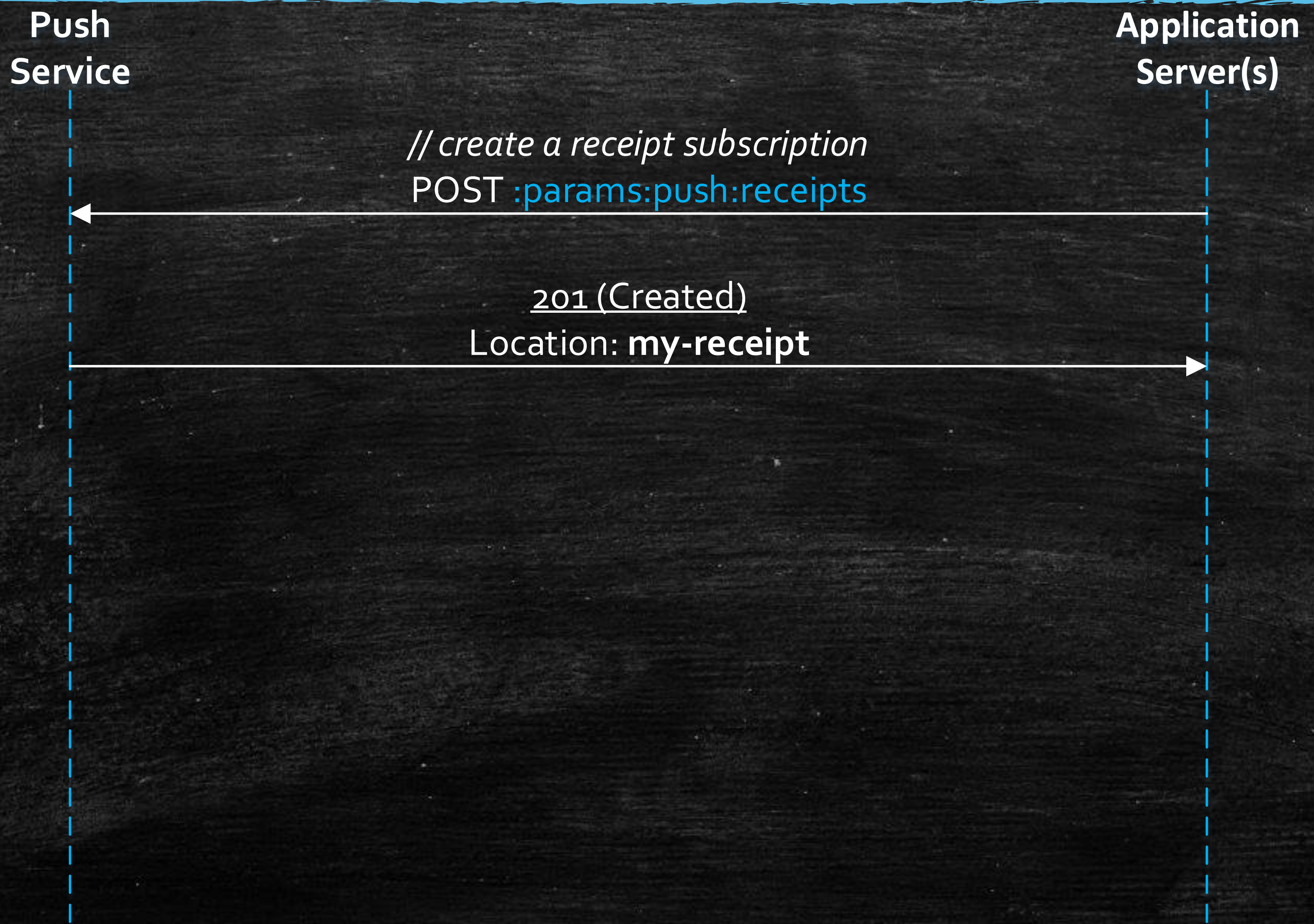


// create message subscription
POST <configured-endpoint>

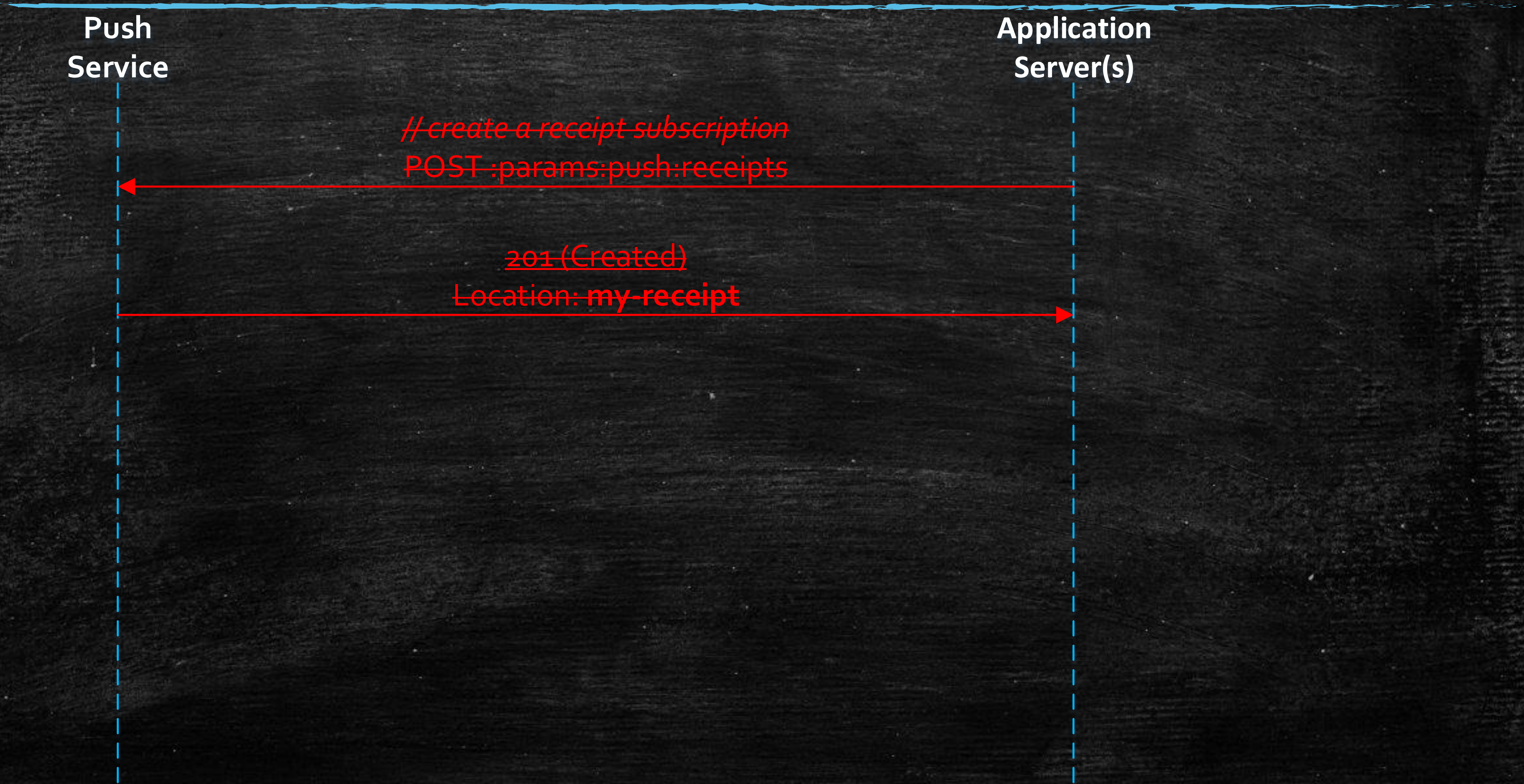
<u>201 (Created)</u>
urn:ietf:params:push
~~urn:ietf:params:push:receipts~~
urn:ietf:params:push:set
Location : **my-messages**

Distribute :params:push ~~*and* :params:push:receipts~~

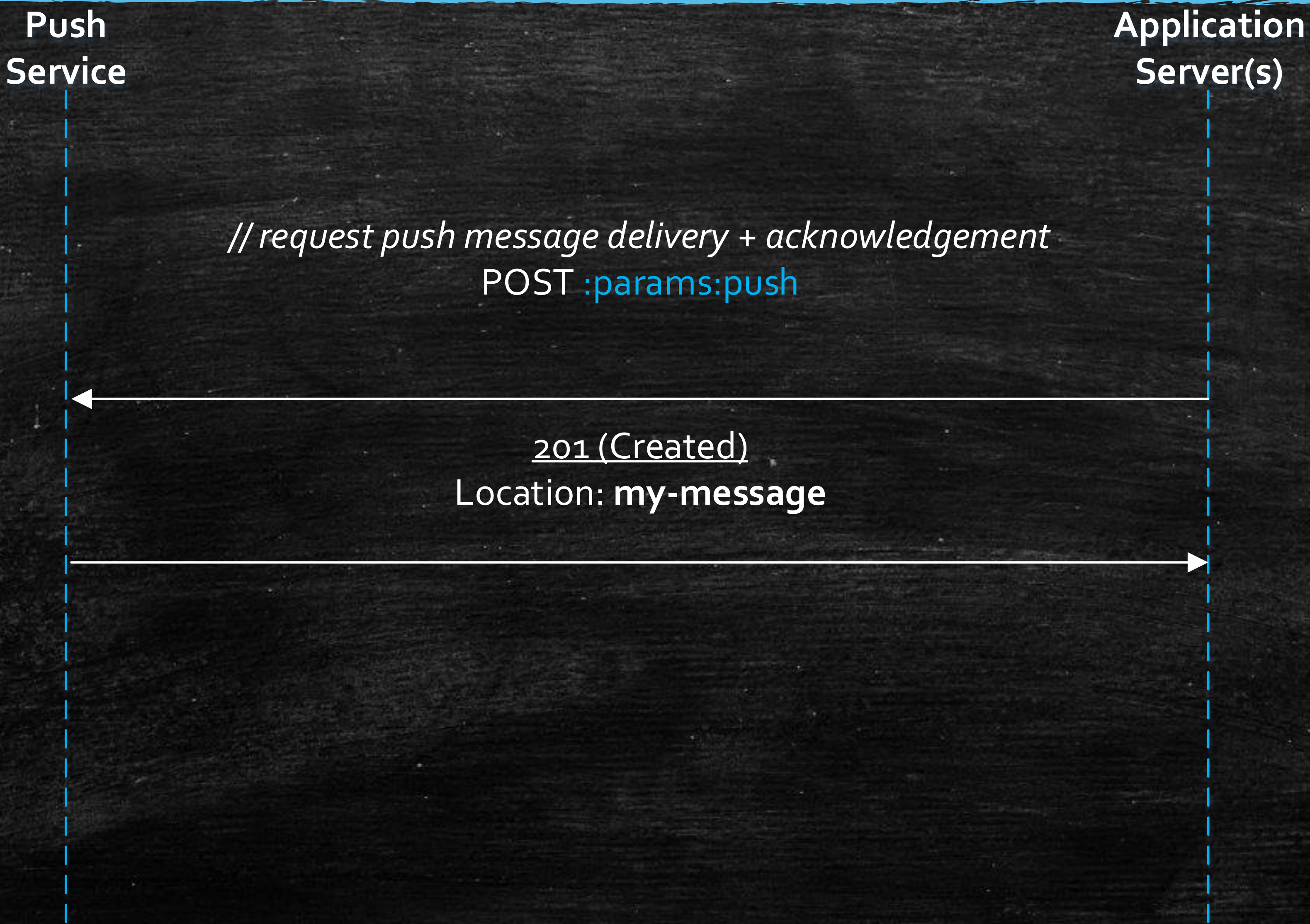
Subscribing to Receipts (Before)



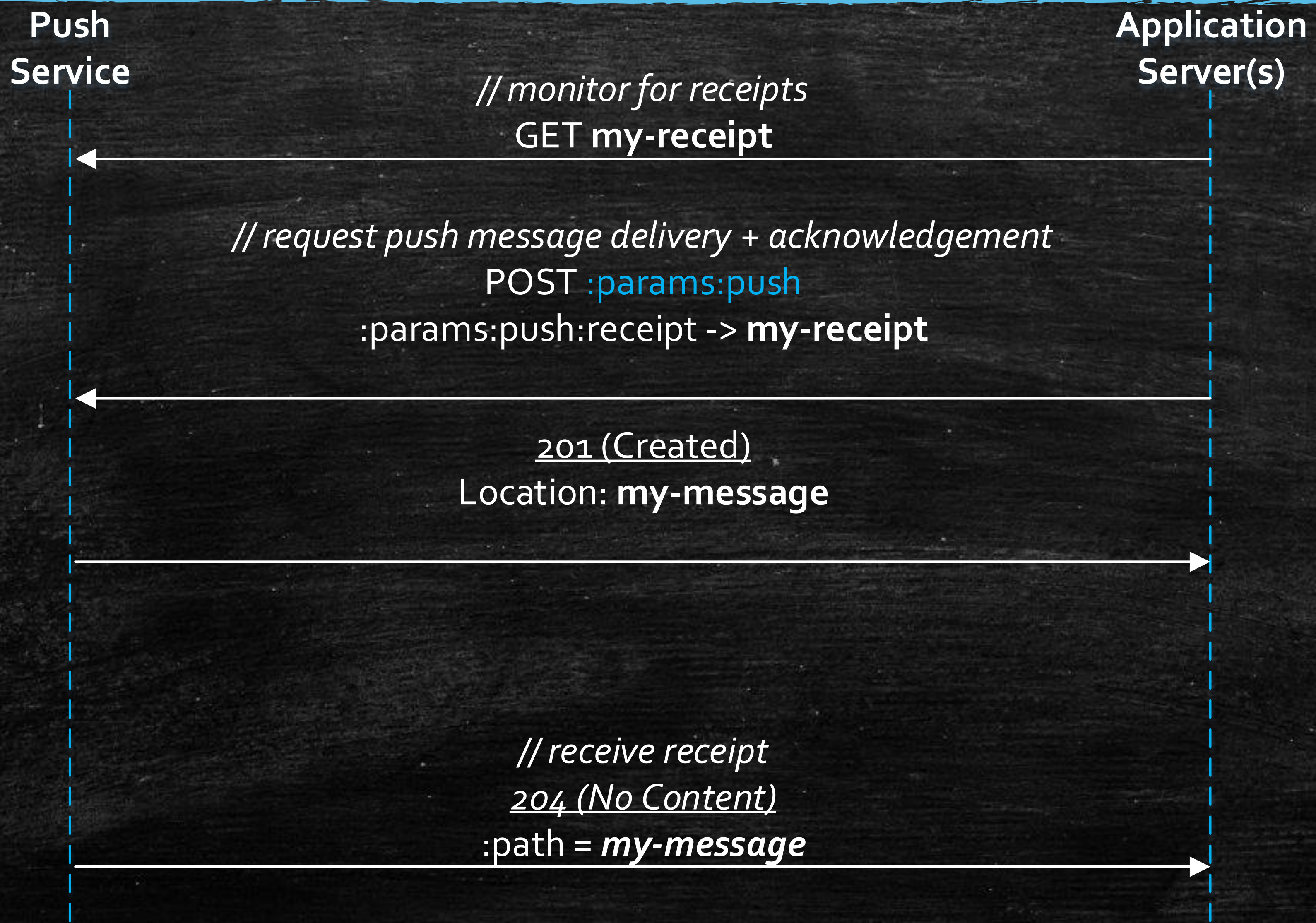
Subscribing to Receipts (After)


Push Service
Application Server(s)
~~// create a receipt subscription~~
~~POST :params:push:receipts~~
~~201 (Created)~~
~~Location: **my-receipt**~~

Publishing without Receipts (unchanged)



Publishing with Receipts (Before)



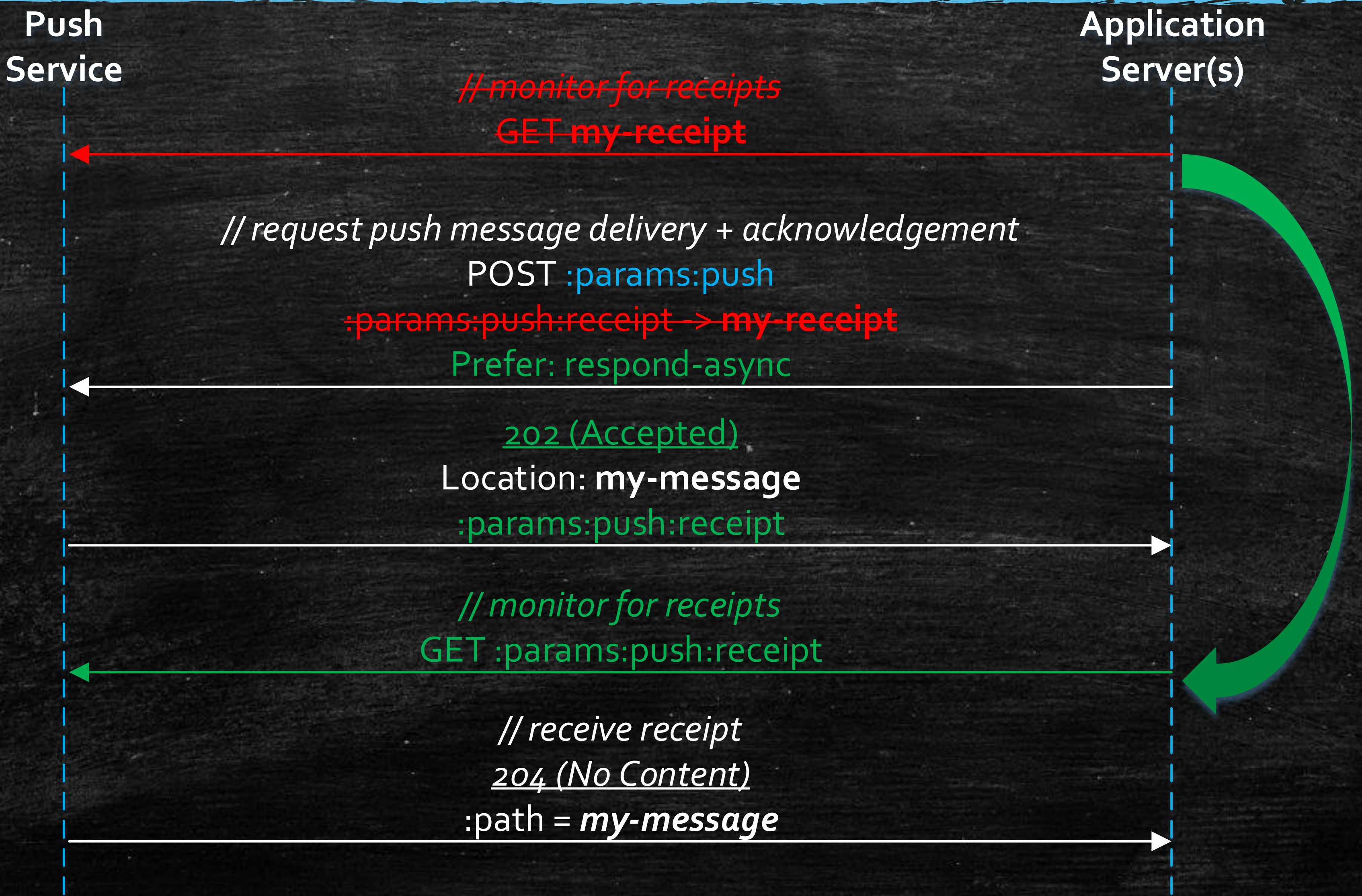
// monitor for receipts
GET **my-receipt**

// request push message delivery + acknowledgement
POST :params:push
:params:push:receipt -> **my-receipt**

<u>201 (Created)</u>
Location: **my-message**

// receive receipt
<u>204 (No Content)</u>
:path = ***my-message***

Publishing with Receipts (After)



Requesting receipts with Prefer: respond-async

```
POST :params:push
Prefer: respond-async
```

The "respond-async" preference indicates that the client prefers the server to respond asynchronously to a response. For instance, in the case when the length of time it takes to generate a response will exceed some arbitrary threshold established by the server, **the server can honor the "respond-async" preference by returning a 202 (Accepted) response**.

Monitoring Status of receipts with :params:push:receipt

202 (Accepted)
Location: **my-message**
:params:push:receipt

The 202 (Accepted) status code indicates that the request has been accepted for processing, but the processing has not been completed … The representation sent with this response ought to describe the request's current status and **point to (or embed) a status monitor that can provide the user with an estimate of when the request will be fulfilled**.

urn:ietf:params:push:receipt

- The urn:ietf:params:push link relation corresponds to and provides write access to a specific subscription for a specific User Agent.

Assuming that the User Agent only distributes that params:push link to one Application Server …

- The Push Service can return the same urn:ietf:params:push:receipt for message delivery requests from an Application Server to the same urn:ietf:params:push.
- The Application Server monitors one resource for receipts from a specific subscription for a specific User Agent.

“What would the :receipt link identify?” - Martin Thomson

This is a resource that the application server would monitor for receipts. But that has the same problem we had with subscriptions that led to the creation of subscription sets, namely that the application server has to make a request for every push message it wants to track.

If this were to act like a subscription set rather than a subscription , … this would be better. The only question then is how an application server causes its receipts to be correlated. The obvious answer is to create the subscription set in an initial request and use the inclusion of the link relation in the push message request as an indication that the application server wants to link receipts to an existing subscription.