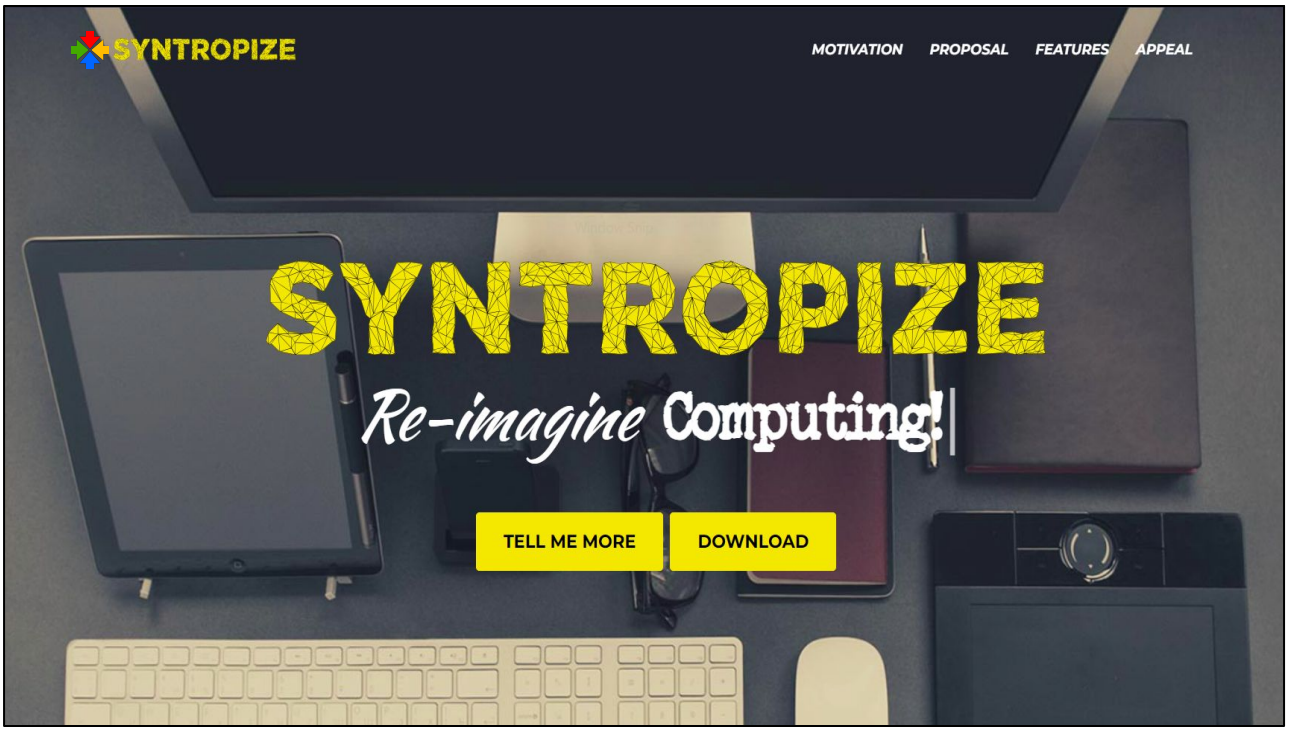
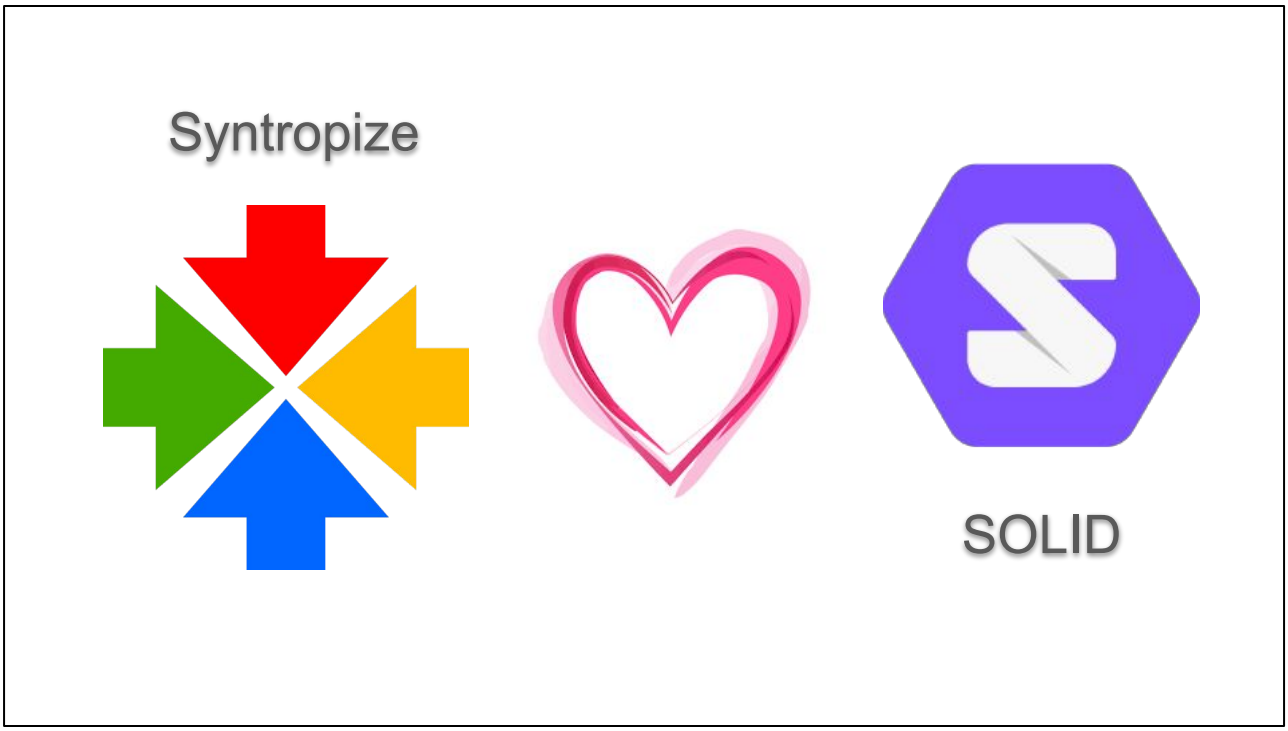
Per Resource Events
&
Solid-PREP

Rahul Gupta

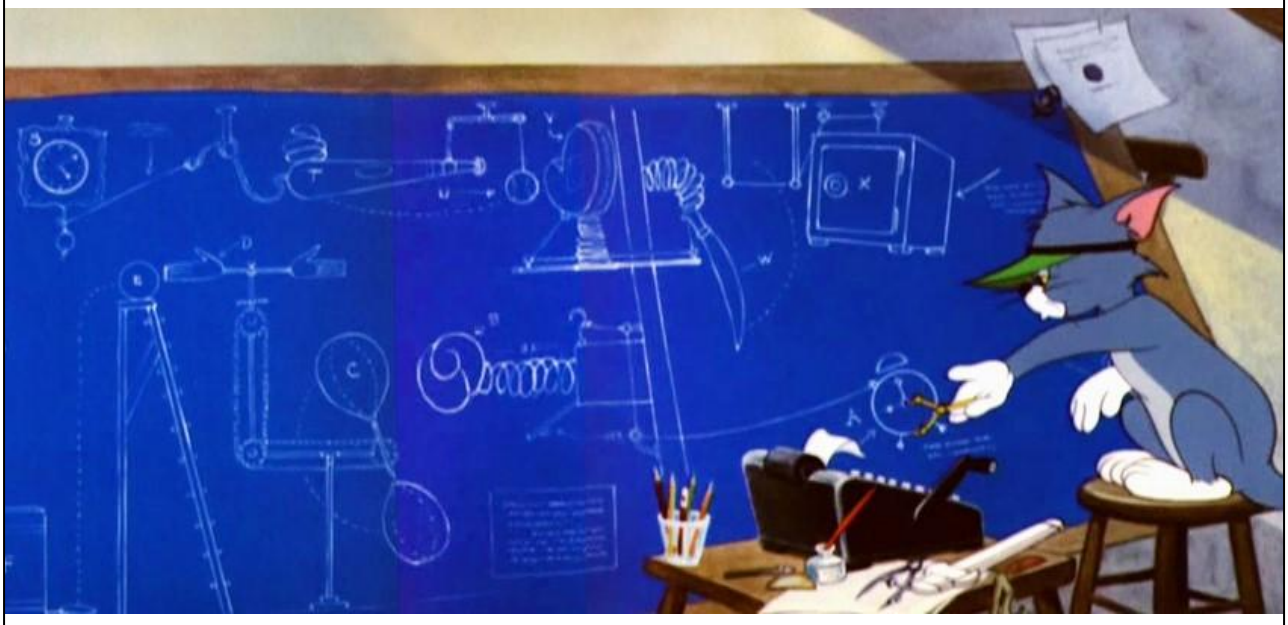
2nd Solid Symposium, Leuven



I will start with some shameless self promotion. I am developing a new operating environment, with the teeny-tiny ambition to replace all desktops and homescreens, called Syntropize. I believe the desktop was a historical mistake, but that is a whole other discussion.

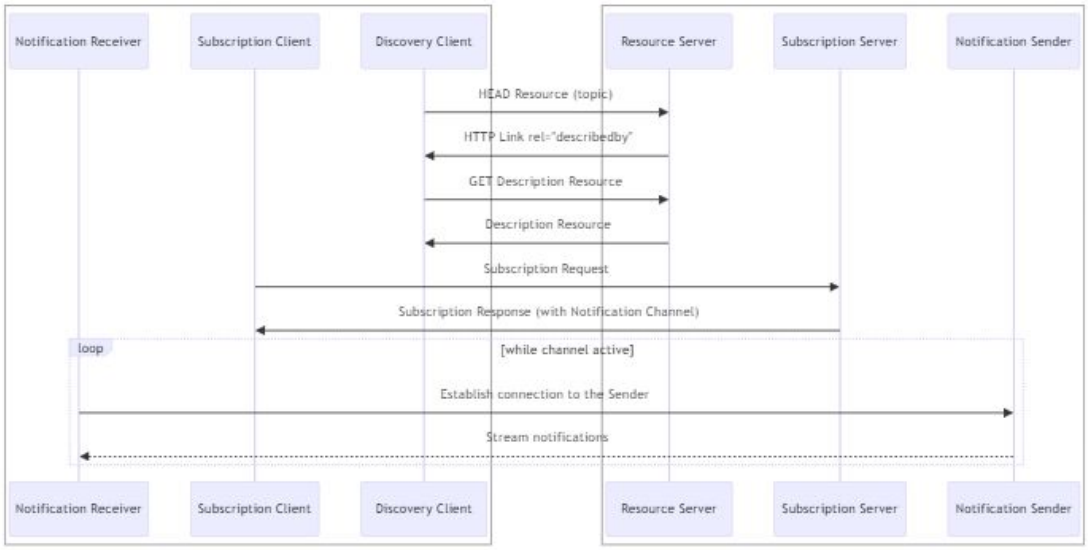


Syntropize allows users to use Solid PODS to store data online. I do not need to explain to this crowd why Solid is a great choice. And I want notifications to get real time update from my pods. However, working with Solid notifications sometimes feels like…



This! Sorry that slide is for the better mousetrap presentation!

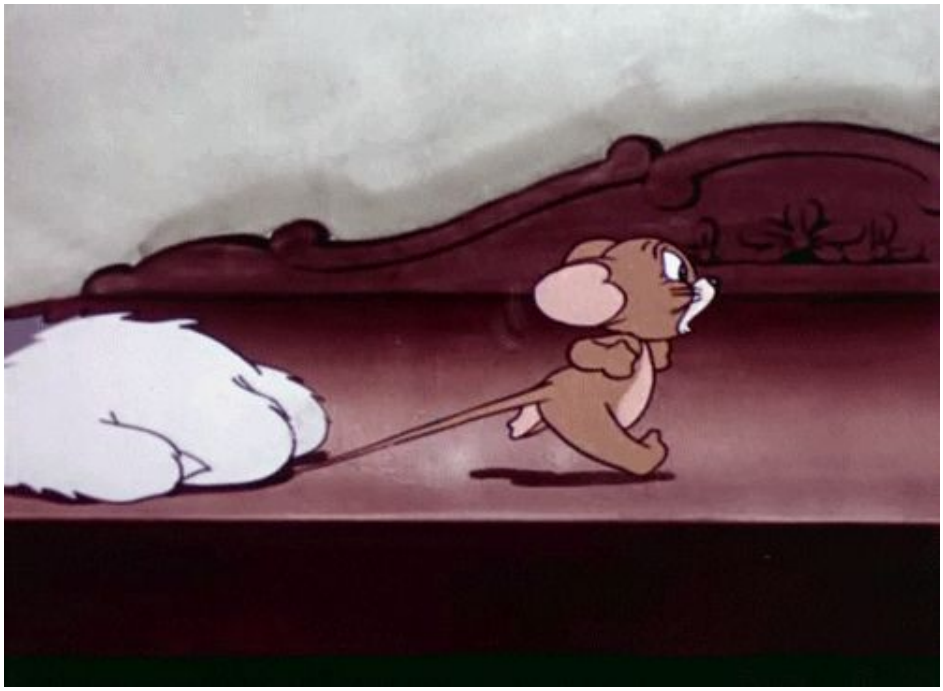
Solid Notifications Protocol



I meant like this! Pavlik has already done a wonderful job of explaining Solid Notifications Protocol.

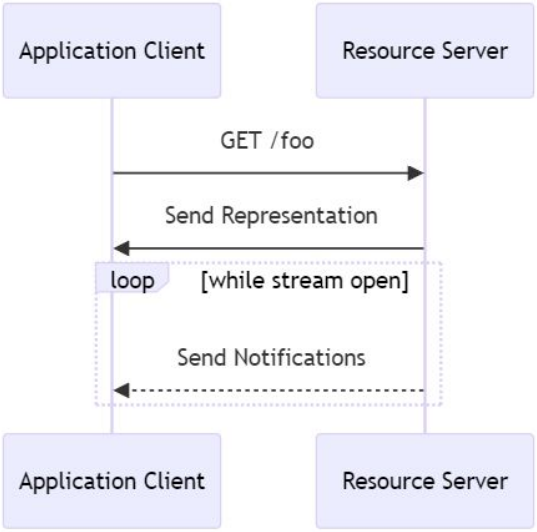
Even then, when we put all of it together, as a co-author of this specification, this is quite overwhelming.

And this diagram does not even account for the fact that the entities on the server side and on the client side need to co-ordinate.



Sometime we want something very simple…

Per Resource Events



This is where the Per Resource Events protocol comes in. A very simple notification flow… We have one request to a resource and we get one response with first the representation and then notifications. We can omit the representation conditionally or completely as we will see later.

Ordinary Request

```
GET /foo HTTP/1.1
Host: example.org
Accept: text/plain
```

If this is what a standard HTTP Request to Solid resource looks like…

Per Resource Events Request

```
GET /foo HTTP/1.1
Host: example.org
Accept: text/plain
Accept-Events: “prep”
```

To get PREP notifications we add a single header.

(The Accept-Events header is completely general. It can be used to negotiate other notification protocols that use different fields, different framing and different semantics!)

Ordinary Response

```
HTTP/1.1 200 OK
Last-Modified: Sat, 1 April 2023 10:11:12 GMT
Transfer-Encoding: chunked
ETag: 1234abcd
Content-Type: text/plain

Hello World!
```

Suppose this were the ordinary response

Response/1

```
HTTP/1.1 200 OK
Last-Modified: Sat, 1 April 2023 10:11:12 GMT
Transfer-Encoding: chunked
ETag: 1234abcd
~~Content-Type: text/plain~~
Vary: Accept-Events
Accept-Events: PREP; accept=message/rfc822
Events: protocol=PREP, status=200
Content-Type: multipart/mixed; boundary=MAIN-SEPARATOR

--MAIN SEPARATOR
Content-Type: text/plain

Hello World!
--MAIN SEPARATOR
```

I'll show you the response in 2 parts:

1. How the representation sent is changed
2. How notifications are sent

The response with notifications adds three headers

- The Events header: which tells the client what type of notifications have been negotiated
- The Vary header: which says the response changed because of the events header
- And also an Accept-Events header: which tells what notifications might be negotiated the next time

The notifications response replaces the content type with multipart/mixed and then moves the representation
in the first part of the multipart.

Following that, the second part is the notifications response of type multipart/digest. Each digest part is a notification.

Response/2

```
Content-Type: multipart/digest; boundary=MESSAGE-SEPARATORC

--MESSAGE-SEPARATOR

Method: PUT
Date: Sat, 1 April 2023 10:11:12 GMT
Event-ID: 1234
E-tag: abc123
--MESSAGE-SEPARATOR

Method: DELETE
Date: Sat, 1 April 2023 10:11:12 GMT
Event-ID: 1234
--MESSAGE-SEPARATOR--
--MAIN-SEPARATOR--
```

Here we see the notifications.

The first one is for a PUT request that re-writes the content of the resource. Had we added a delta parameter in the request, the notification would have body

The second one is for a DELETE request. Since the resource ceases to exist, after the notification the notification response stream is closed as per the rules of multipart in RFC2046.

Request

Notifications Only

```
GET /foo HTTP/1.1
Host: example.org
Accept: text/plain
Accept-Events: “prep”
Last-Event-ID: *
```

And suppose you want notifications without the representation.

The client can specify the last-event-id (repurposed from SSE) as star. This tells server to send notification from whatever ID it generates next. The response now will be just multipart/digest part.

Request

Negotiating the notification format

```
GET /foo HTTP/1.1
Host: example.org
Accept: text/n3
Accept-Events: “prep”; accept=application/ld+json
```

I will now show you a couple of typical request variations.

PREP allows clients to negotiate the content-type of notifications. And it reuses parameters with the same names as HTTP fields to do so.

In Solid, we often want notifications from RDF resources. So, here we are negotiating notifications in everybody’s favourite media type - JSON LD or shall I say LD plus JSON?

Solid-PREP

- Companion specification for notifications format from Solid/Linked Data resources
  - https://github.com/solid/solid-prep
- Shares message semantics (formats, triggers etc) with Solid Notifications for RDF resources

Solid-PREP is a companion specification that specifies the notifications format from Solid/Linked Data resources in Solid PODS. Our goal is to ensure that message format for Solid Notifications and PREP are identical, the client gets the same notification, no matter what protocol they choose to get notifications using.

Let me remind you that PREP can be used on non-RDF resources as well, in which we use can content negotiate another notification format suitable for that resource.

Comparison with Solid Notifications

PREP

- Single request
- Uses Streaming HTTP Only
- Single Resource Notifications

Solid Notifications

- Two requests (Requires Discovery)
- Supports multiple channel types
- Supports Notifications from multiple resources

Let me emphasize that PREP is a complementary notifications mechanism to Solid Notifications. The two address the needs for different audiences.

If you need to send notifications of a different protocol such as websockets or webhooks, you are still going to use Solid Notifications. If you want notifications from multiple solid resources over a single connection again you would want to use Solid Notifications.

However, if you are looking to get notifications from one resource at a time and want to connection with a single request, PREP is great option for you.

Implementation

- Available as Express Middleware
  - Express-Accept-Events
  - Express-PREP
  - Express-Events-Negotiate (for negotiating other protocols)
- (partially) Implemented in Node Solid Server
- Green Light from the Community Solid Server

Demo!

Questions