Solid Pods:
A Deep Dive

Michiel de Jong
michieldj@inrupt.com

What is Solid?

- A set of specifications
- Protocol specifications define a language
- Server / API specifications define a behaviour
- Client-client specs for interop between apps
- Solid needs all three!

The goal

- Separate apps from data
- Allow the user to choose an app
- Independent of their pod provider
- Independent of which app they previously used
- Allow the user to freely choose any pod provider

Deep dive

- Main part: protocol spec + API spec of pod server
- The web: URLs, DNS, HTTP (1.1 https/tls http/2)
- Read-write web: LDP BasicContainers as RDF
- Basic RDF conneg (Turtle / JSON-LD)
- WebId-OIDC bearer tokens, DNS-based identity
- ACL docs in RDF (Web Access Control / WAC)

How read/write works

- Slashes in URL path are nodes in a tree, `ls`
- Possibly serve a html UI or html folder index
- PUT with mkdir-p
- POST
- DELETE file, DELETE (empty) folder
- If-Match, If-None-Match: *

Web Access Control

- ACL doc (in RDF, of course!)
- Type `acl:Authorization`
- `acl:agent`, `acl:agentGroup` statement
- `acl:origin` statement (optional)
- `acl:mode` (Read, Append, Write, Control)

Finding the ACL doc

- Through Link response header (`acl:accessTo`)
- Or (parent) container (then `acl:default`)

What the modes mean

- `acl:Control` on X means read+write on ACL of X
- `acl:Read`: read body / containment triples
- `acl:Append`: add to body / containment triples
- `acl:Write`: create or modify a resource
- mkdir-p means creating requires parent append
- Delete requires `acl:Write` on resource + parent

Authorization: Bearer

- It’s complicated :)
- The token proves user controls the WebId
- Is specific to an Origin (even if no Origin header)
- An IDP allows a web app to obtain the token
- IDPs should allow WebId-TLS as a login method

acl:trustedApp

- To override need for `acl:origin`
- In the WebId profile of a user with `acl:Control`
- Publicly
- Will probably soon be replaced

WebSockets-pubsub

- Discover through `Updates-Via` header
- Send `pub <URL>`
- Receive `ack <URL>`
- Receive `pub <URL>`

WAC-Allow

- List which access modes the current user has
- List which access modes of anonymous user

PATCH

- sparql-update
- Only INSERT / DELETE, no WHERE
- May require just `acl:Append` or full `acl:Write`

RDF merge

- GET merge of all RDF docs in a container
- aka “globbing”
- At risk for December version

GET with SPARQL

- Filter triples from a single RDF doc
- At risk for December version

Minimal pod contents

- Profile doc (at user's WebId! links pod with IDP)
- public/private type index
- LDN inbox

Client-client spec

- Which vocabs to use (plural!)
- Which statements to include
- Always start at the profile doc
- Discoverability through links
- Which URL can state what?

```
</profile/card#me> a          foaf:Person .
</profile/card#me> a          schema:Person .
</profile/card#me> foaf:name "John Doe" .
```

Linking to your pod

Say your pod is at `/pod`, with the LDN inbox at `/pod/inbox/`, to link from your identity to your pod:

```
</profile/card#me> solid:account </pod> .
</profile/card#me> pim:storage   </pod> .
</profile/card#me> ldp:inbox     </pod/inbox/> .
```

Preferences

To publish some of your generic preferences to apps, use:

```
</profile/card#me> pim:preferencesFile    </settings/prefs.ttl> .
</profile/card#me> solid:publicTypeIndex  </settings/publicTypeIndex.ttl> .
</profile/card#me> solid:privateTypeIndex </settings/privateTypeIndex.ttl> .
```

Address Book

You can create an addressbook containing persons and groups, by adding triples to RDF documents on your pod. To create an addressbook, create a document for it, e.g., `/address-book/index.ttl`, and add the following triples to that document:

```
</address-book/index.ttl#this> a          vcard:AddressBook .
</address-book/index.ttl#this> dc:title  "New address Book" .
</address-book/index.ttl#this> acl:owner </profile/card#me> .
```

You can create separate documents for the people index and for the groups index, as long as you link to those from the main `/address-book/index.ttl` document in the following ways:

```
</address-book/index.ttl#this> vcard:nameEmailIndex </address-book/peopleIndex.
</address-book/index.ttl#this> vcard:groupIndex     </address-book/groupIndex.t
```

To indicate that a person `/johnDoe.ttl` with full name "John Doe" is in addressbook `/address-book/index.ttl`, add the following triples: