# Most simple example for HEART based sharing

Note: We have intentionally reduced this use case and example to represent the simplest possible example.

Use Case: Patient Alice wishes to share her medical record with Dr. Erica (outside of current organization) who she was referred to. (Proactive Sharing)

Resource: Alice wishes to exchange her list of medications. (FHIR resource MedicationStatement)

## Assumptions:

* Alice’s open-id email is AliceSamplePatient@gmail.com. Her PatientID where her data is stored is ‘123’
* Dr. Erica has an OpenID Provider with an email address, which Alice’s AS is willing to be a relying party for. Dr. Erica’s email address is email address is DrErica@gmail.com
* Alice has already created a consent policy for this transaction which specifies
  + She wishes to share her list of medications
  + The scope is Read only
  + She wants to share with Dr. Erica, as identified by DrErica@gmail.com
  + The purpose is HealthCare Data Sharing
  + She has authorized sharing for 11/1/2016-10/31/2017
* Dr. Erica has been notified that Alice’s clinical data is available, including
  + Alice’s PatientId
    - This is an opportunity for HEART to profile how we will notify the user we are sharing with.
  + The End-point where Alice’s data resides: rs.example.com
* The Client has client credentials before this workflow starts.
* The Resource Server has previously registered available Resources for Alice. There is a resource set consisting of Alice’s records for her medication list. That resource\_Set\_ID was registered as ‘ABC’. The friendly name is ‘Medications I Take’
* The RS has already been authorized at the AS before this workflow starts, i.e. The RS has a PAT. (Presumably this happens when Alice makes the connection from the AS to the RS.)
  + The UMA protection API token (PAT) is an OAuth access token that functions as a pairwise pseudonym associating Alice-at-RS and Alice-at-AS. The PAT allows the RS to refer to Alice every time it registers resource sets and requested permissions at the AS.
* The AS endpoint is defined as as.example.com

## Now the workflow starts.

1. The client requests Alice’s medication list from RS
   1. Alice’s patientID and RS endpoint are known. (The process of sharing informs Dr. Erica where to find the resource.)
      1. RS endpoint url
      2. PatientID

GET [Server EndPoint]/[FHIR resource type]/'patient='{PatientId}

Ex: GET /rs.example.com/MedicationStatement?patient=123

1. RS is requesting permission, on behalf of the client, at the AS for attempted access

POST /host/rsrc\_uri HTTP/1.1

Content-Type: application/json

Host: as.example.com

Authorization: Bearer 204c69636b6c69 --This is the PAT, AS provides the PAT to the RS when the RS logs into the AS with the RS credentials

{

"resource\_set\_id": "ABC",

"scopes": [

"read"

]

}

Note: All the internal mapping that the RS server needs to do to convert the API call to the message back to the AS.

1. AS returns the permission ticket to RS

HTTP/1.1 201 Created

Content-Type: application/json

...

{

"ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de"

}

1. RS returns permission ticket and AS URI to client (with code 401).

HTTP/1.1 401 Unauthorized

WWW-Authenticate: UMA realm="example",

as\_uri="https://as.example.com/rpt\_uri",

ticket="016f84e8-f9b9-11e0-bd6f-0021cc6004de"

...

1. This is where Dr. Erica may need to confirm who she is (either finish creating her account or login)
   1. Dr. Erica needs to sign in (via AS) to get an AAT (Dr. Erica needs to identify herself to the AS.) The AAT is generated and returned. One of the following options must be used:
      1. Local account. If Dr. Erica hadn’t had an account, she would have had to create one.
      2. Federated account. Dr. Erica can use an existing account that Alice’s AS accepts. Our assumption includes **this option**.
   2. The UMA authorization API token (AAT) is an OAuth access token that functions as a pairwise pseudonym associating Erica-at-client and Erica-at-AS. The AAT allows the client to refer to Erica during the process of trying to convince the AS to give Erica access to Alice's resources.
2. The client now has the AS endpoint and ticket. The client requests authorization from the AS, providing the permission ticket sent from the RS.

POST /rpt\_uri HTTP/1.1

Host: as.example.com

Authorization: Bearer jwfLG53^sad$#f -- this is AAT, client got it  
 -- when Dr. Erica consented to AS

...

{

"ticket": "016f84e8-f9b9-11e0-bd6f-0021cc6004de"

}

1. The AS checks requests against Alice’s policy, we assume for this case it is granted. The AS grants authorization and provides RPT (Request Party Token) back to the client.

HTTP/1.1 200 OK

Content-Type: application/json

{

"rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv"

}

Returns RPT to Client to use for authorization.

1. Now the client can issue the request to the RS using the RPT

GET …/ MedicationStatement?patient=’123’

Authorization: Bearer "rpt": "sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv"

Host: rs.example.com

1. The RS introspects the RPT by sending it back up to the AS.

POST /introspect HTTP/1.1

Host: server.example.com

Accept: application/json

Content-Type: application/x-www-form-urlencoded

Authorization: Bearer 23410913-abewfq.123483 -- PAT

token= sbjsbhs(/SSJHBSUSSJHVhjsgvhsgvshgsv -- RPT

Note: In UMA 1, the client is required to introspect. This is changing in UMA2, but for now is required.

AS returns a resource set id and list of granted scopes. Then RS derives status from granted scopes. We will assume for this example it is granted.

HTTP/1.1 200 OK

Content-Type: application/json

Cache-Control: no-store

{

"active": true,

"exp": 1256953732,

"iat": 1256912345,

"permissions": [

{

"resource\_set\_id": "ABC",

"scopes": [

"read"

],

"exp" : 1256953732

}

]

}

1. RS can then return he data to the client as it would normally do in FHIR