Requesting access with Requestor

Sara Volk de Garcia

Center for Translational Data Science,
University of Chicago
July 30, 2025

What is Requestor?

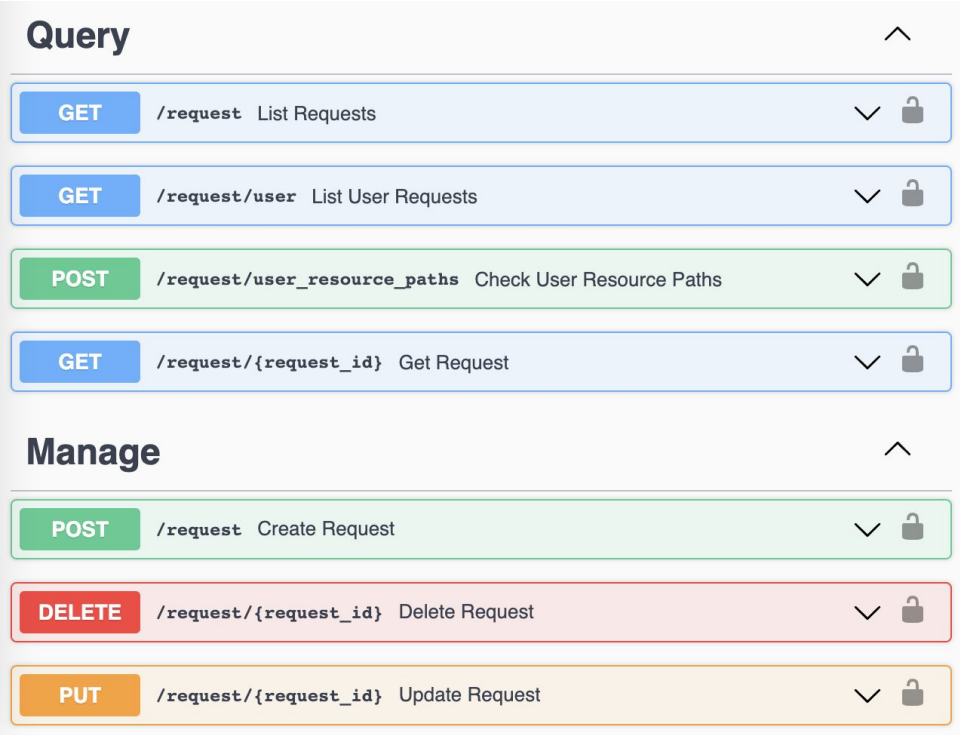
Note: all Requestor APIs have the prefix `/requestor`

Requestor is an API to manage access requests in a scalable and auditable manner.

It allows both granting access and revoking access to resources and policies.

[Link to Requestor GitHub repo]

[Link to Requestor API documentation]



Advantages of Requestor for access management

- (fairly) scalable
- Auditable
- May be integrated with third-party tools to surface the requests in a help-desk format (e.g., Zendesk)
- Allows you to manage (grant and revoke) complex policy combos

Overview

You have a resource for which you want to control access.

1. Enable Requestor in your Gen3 instance
2. Create a policy in the user.yaml that identifies a role (with permissions) and a resource to which users need access
3. (also create a policy that all authenticated users may create Requestor requests)
4. Authenticated users may request access, creating a Requestor request
5. Admins with appropriate permissions may approve (or not) access requests
6. Users with approved requests will have access as defined by the policy

Requestor administrators

Requestor allows a group of “admin” users to manage access requests to specific data.

It allows a process of review by this group of users: the access is only granted if an admin approves the request.

We can restrict who can see existing access requests, and who can create access requests (for themselves or others) in the same way.

```
roles:
  - id: requestor_admin
    permissions:
    - id: requestor_admin_action
      action:
        service: requestor
        method: '*'
  - id: requestor_reader
    permissions:
    - id: requestor_reader_action
      action:
        service: requestor
        method: read
  - id: requestor_creator
    permissions:
    - id: requestor_creator_action
      action:
        service: requestor
        method: create
  - id: requestor_updater
    permissions:
    - id: requestor_updater_action
      action:
        service: requestor
        method: update
  - id: requestor_deleter
    permissions:
    - id: requestor_deleter_action
      action:
        service: requestor
        method: delete
```

```
policies:
  - id: requestor_reader
    role_ids:
    - requestor_reader
    resource_paths:
    - /study
    - /mds_gateway
    - /cedar
  - id: requestor_creator
    role_ids:
    - requestor_creator
    resource_paths:
    - /study
    - /mds_gateway
    - /cedar
    - /workspace
    - /workspace_stride_grants
    - /workspace_stride_credits
  - id: requestor_updater
    role_ids:
    - requestor_updater
    resource_paths:
    - /study
    - /mds_gateway
    - /cedar
    - /workspace
    - /workspace_stride_grants
    - /workspace_stride_credits
  - id: requestor_deleter
    role_ids:
    - requestor_deleter
    resource_paths:
    - /study
    - /mds_gateway
    - /cedar
    - /workspace
    - /workspace_stride_grants
    - /workspace_stride_credits
```

Requestor administrators

Here we have a role: requestor_updater. This person can approve requests

We have a policy that assigns the requestor_updator role to a specific project resource path. A user with this policy would be allowed to update requests for that resource (but not other resources).

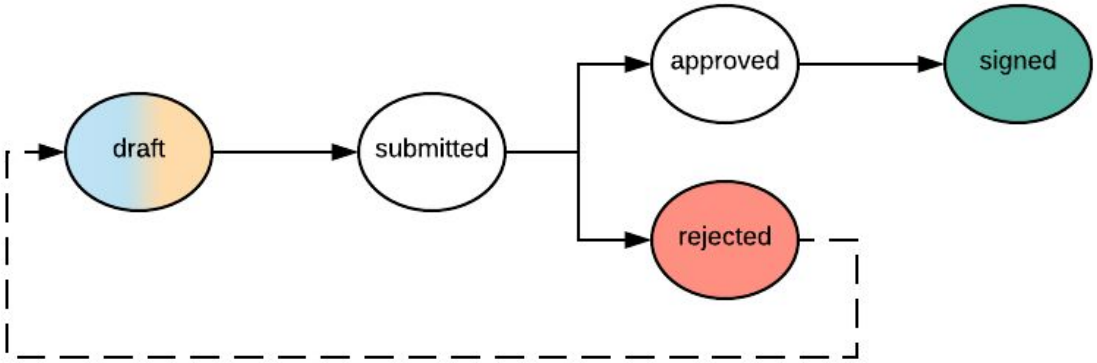
We have a group that allows you to assign the policy (or more than one policy) to specific users in the user.yaml

```
roles:
- id: requestor_updater
  permissions:
  - id: requestor_updater_action
    action:
      service: requestor
      method: update
policies:
- id: PROJ_requestor_updater
  role_ids:
  - requestor_updater
  resource_paths:
  - /programs/PROG/project/PROJ
groups:
- name: PROJ_access_request_admins
  policies:
  - PROJ_requestor_updater
  users:
  - me@uchicago.edu
```

Access request workflow

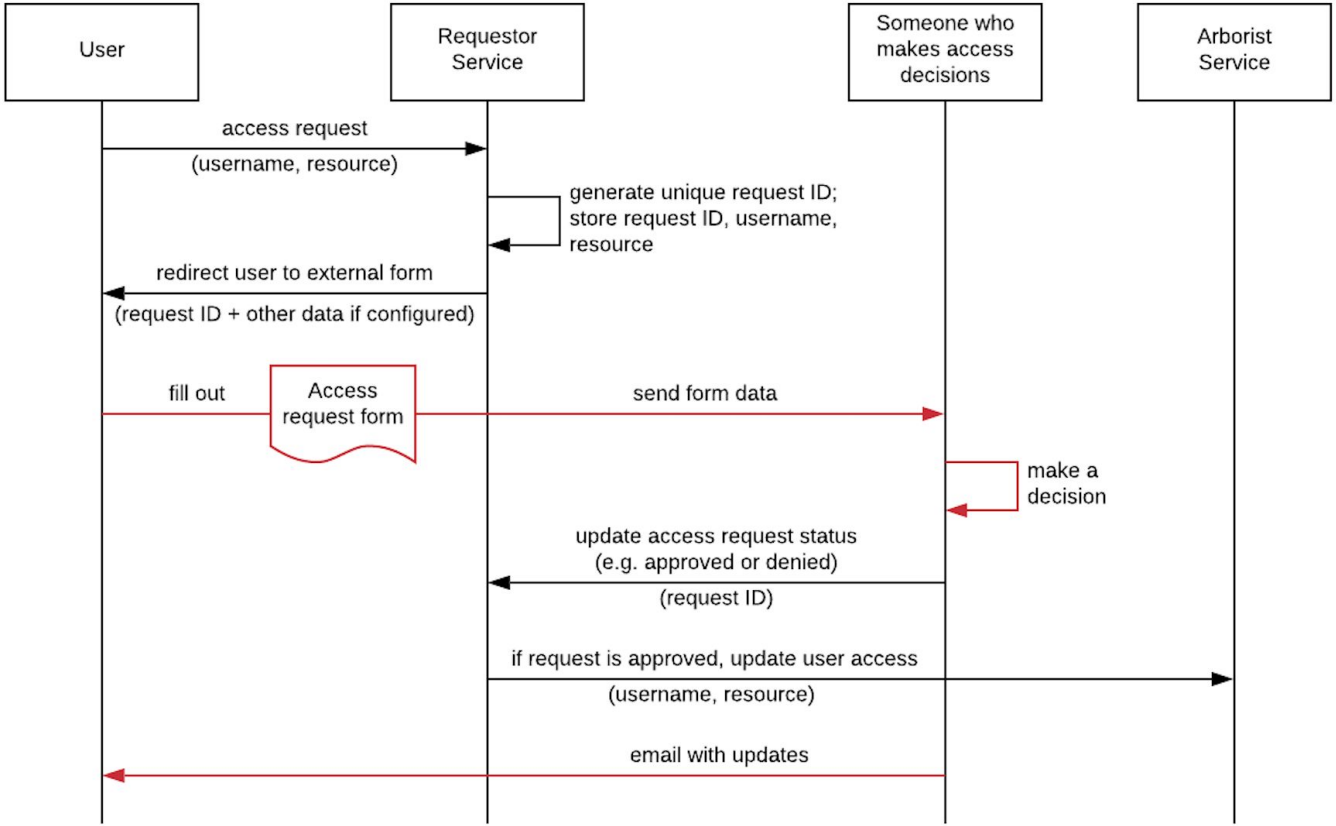
Requestor is configurable and can match the implementing entity's access request workflow.

Example: the Requestor workflow for access to workspaces in the HEAL project. Access is granted when an access request status is moved to "signed".



Flexible permissions

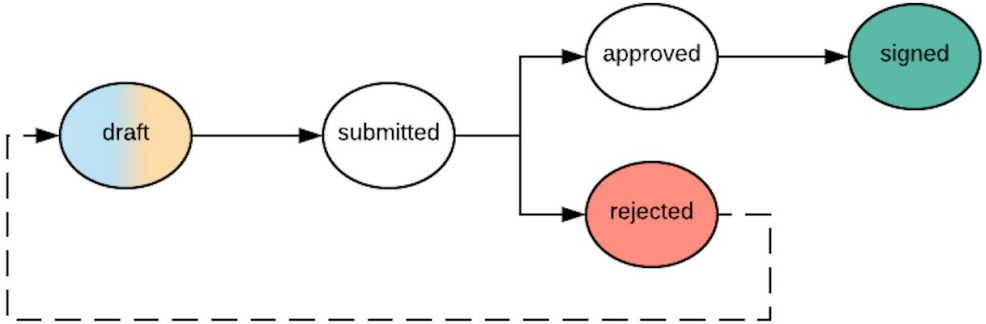
- Requestor allows granting access to a policy. This means we can grant any kind of access (write access, workspace access, etc - anything that’s in the user.yaml).
- It also allows revoking access to a policy.



High level Requestor flow example

Red: not included in the Gen3 codebase

Requestor Statuses



arrows are informational only. there is no logic to enforce order of statuses

the number and names of statuses are configurable

INITIAL_STATUS

access requests are created with this status.

FINAL_STATUSES

access requests cannot be updated anymore, but users are not blocked from requesting access again. a new request ID would be generated.

use case: a request is rejected but the user wants to try again

UPDATE_ACCESS_STATUSES

the user is granted access.

there is no logic to block users from requesting access again (this can be done on the client side) or to prevent updating access requests - unless also configured as a FINAL_STATUS.

DRAFT_STATUSES

users are not blocked from requesting access again. the same request ID would be used but actions (such as redirect) would be triggered again.

use case (NIAID): a user clicks the "request access" button but does not submit the form

other statuses

users are blocked from requesting access again: users can only request access to a resource once at a time.

Using Requestor with Helm deployment

[See how Requestor is enabled in HEAL values.yaml]

[Default configuration for Requestor in Gen3 Helm (Requestor values.yaml)]

Hello!

I am Conrad Leonard

Technical Lead in the Human Genome Informatics program at Australian BioCommons

You can contact me at conrad@biocommons.org.au

Access Requests with REMS

Australian BioCommons

Data Governance Terms

Data Applicant — The researcher. Requests data to support research question

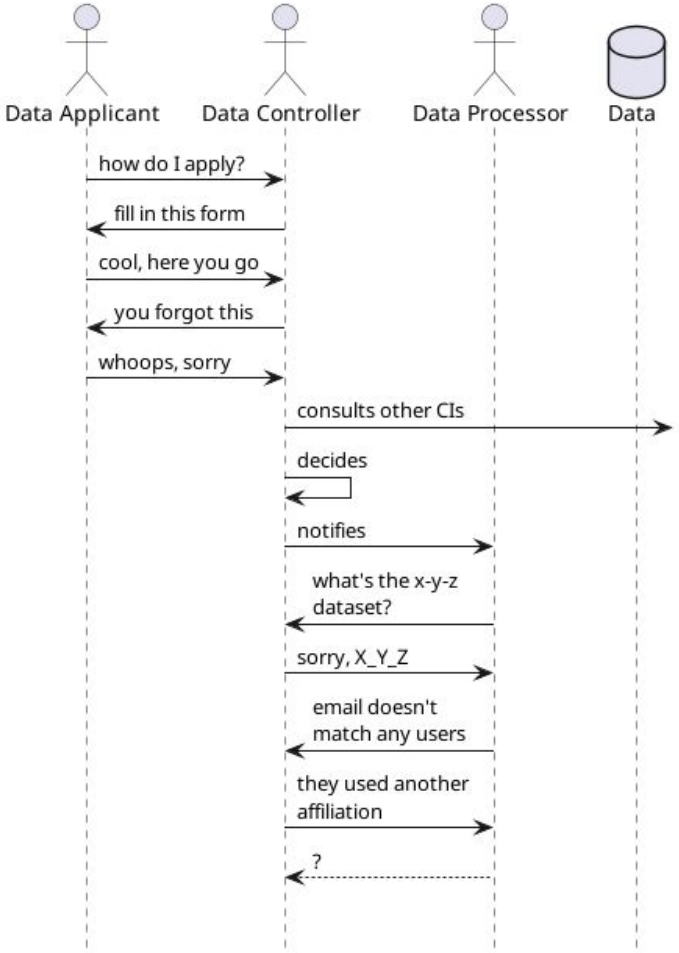
Data Controller — The decision maker. Approves / rejects / revokes access. Sometimes aka ‘steward’ or ‘owner’

Data Processor — Provides services and executes actions based on controller’s instructions e.g. ensures data security

Tier 0 — Public access (anonymous)

Tier 1 — Registered access (logged in)

Tier 2 — Controlled access (approval required)



Australian BioCommons

Resource **E**ntitlement **M**anagement **S**ystem

- Software out of Finland CSC used for access grant management of research datasets
- Supported in research data infrastructure community — e.g. the GDI program, ELIXIR Finland, DKFZ cloud, CKAN extension
- Mature project & actively maintained in GitHub
- Rich API and event notifications — most functionality has an endpoint and the bundled UI is simply an API client

Resource Entitlement Management System

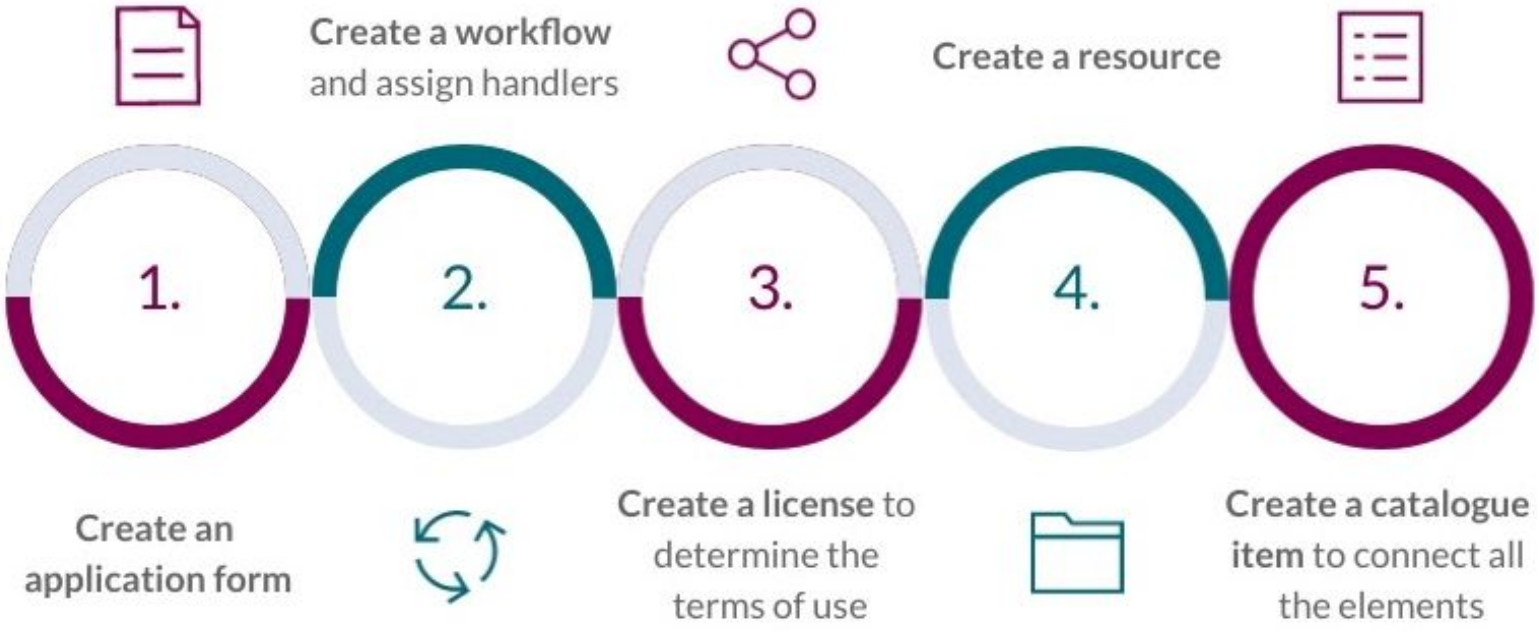
Benefit to the Data Controller

- A dynamic way to handle applications and manage access rights
- An audit trail of all committed actions
- Customisable application process
- Automation: Some functions can be automated (approve/reject) to reduce unnecessary workload.

Benefit to the Applicant

- Use existing institution user account for login
- Follow the application process easily
- See all applications in one place and check their state easily
- Add multiple applicants to one application to streamline the process

REMS — Data Controller User Journey



Australian BioCommons

REMS — Data Applicant User Journey

1. Login

Use the identification method your institution recommends

2. Licenses

Accept the terms of use and add your group members to the application

3. Submit application

Send your application to the Data Access Committee (DAC) for a review

4. Review

The DAC reviews your application and approves or rejects it

5. Access rights

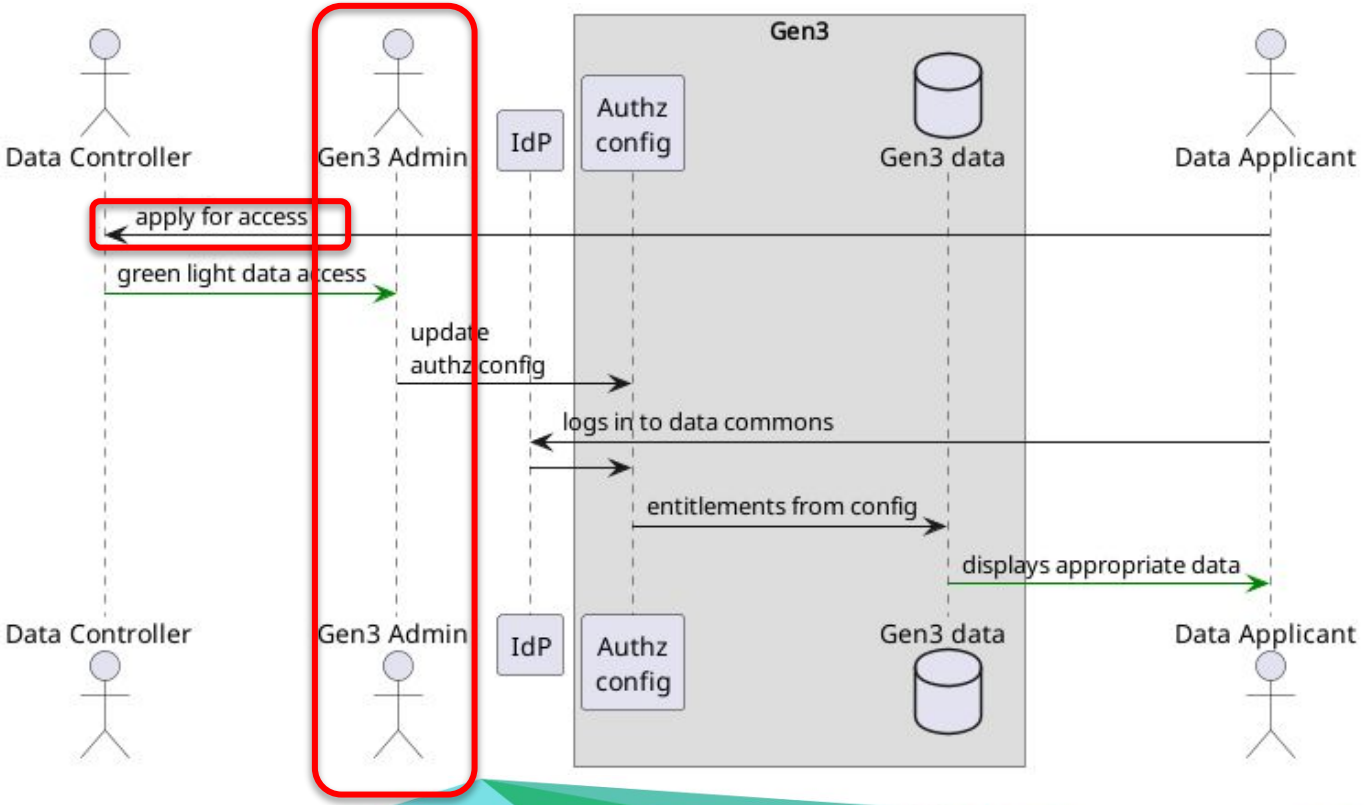
Once the DAC grants you access rights, you can start your research

Australian BioCommons

Data Access Control in Gen3

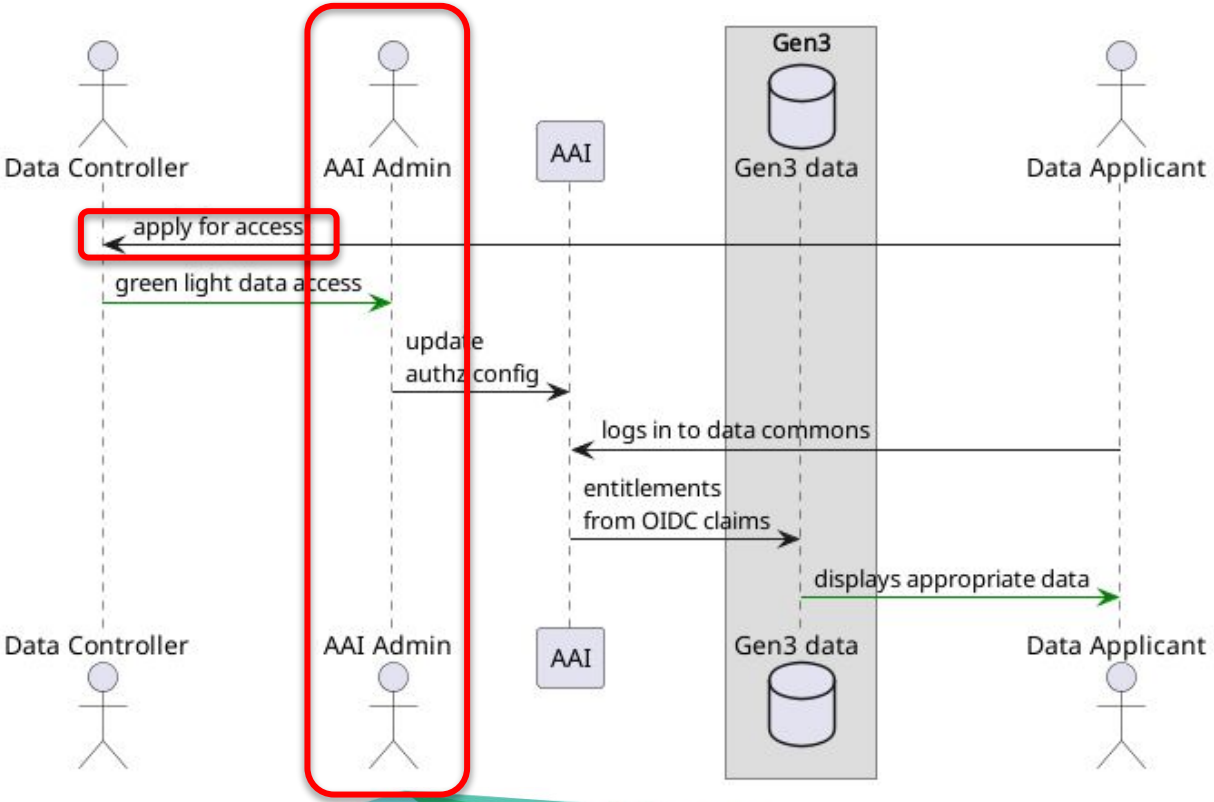
Australian BioCommons

Config-based (o)



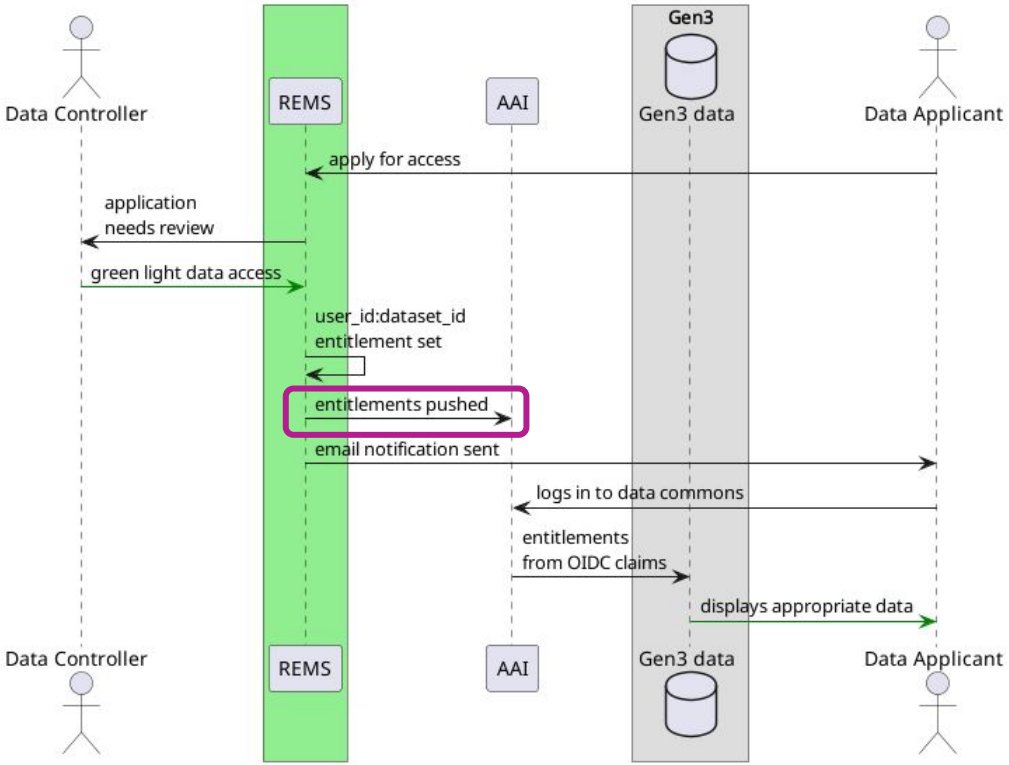
Australian BioCommons

With OIDC (i)



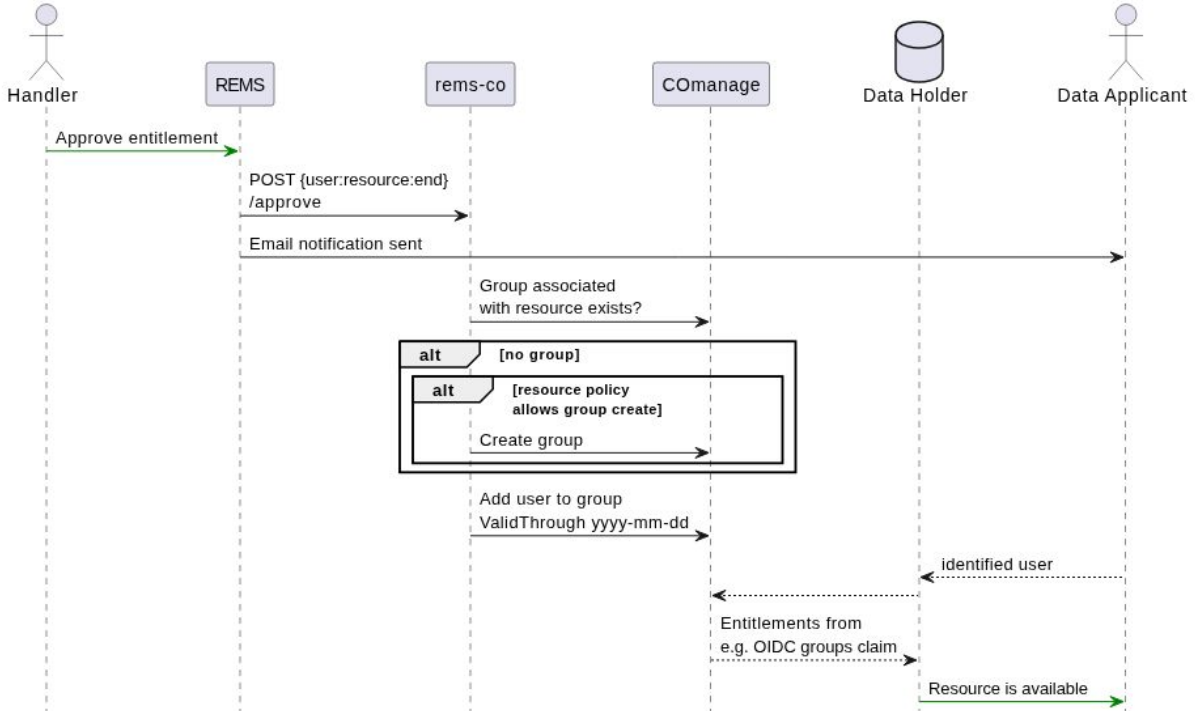
Australian BioCommons

With REMS (ii)



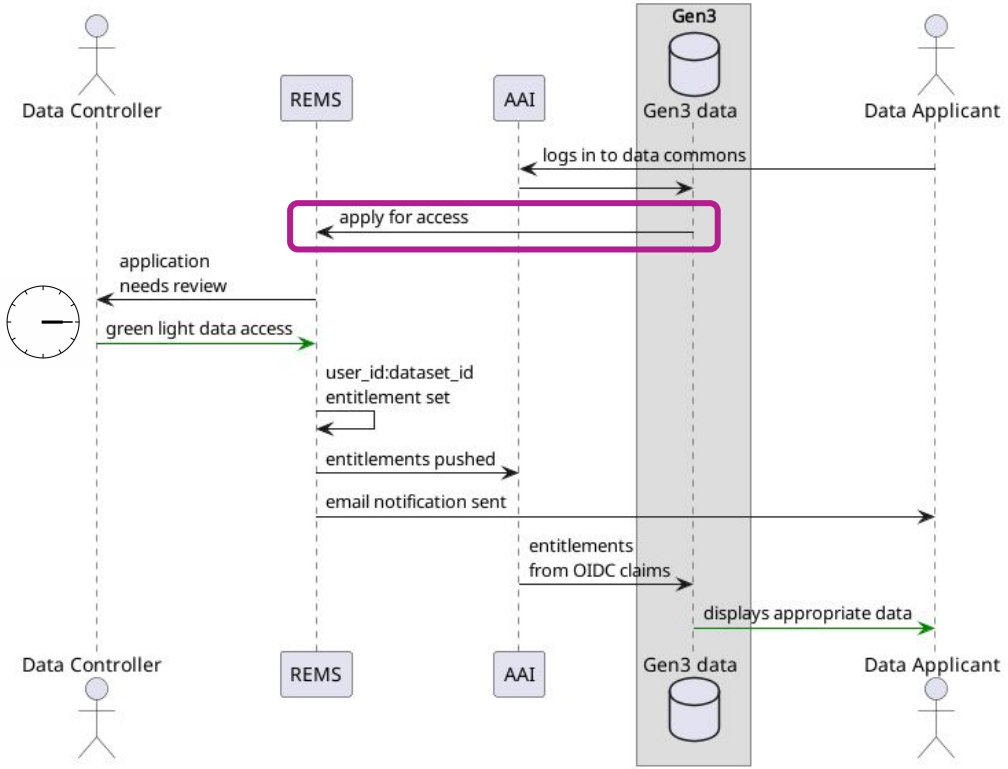
Australian BioCommons

Triggered entitlement push (example)



Australian BioCommons

With... requestor ? (iii)



Australian BioCommons

Thanks!

Any questions?

You can email me at: conrad@biocommons.org.au