Egeria Server Author UI

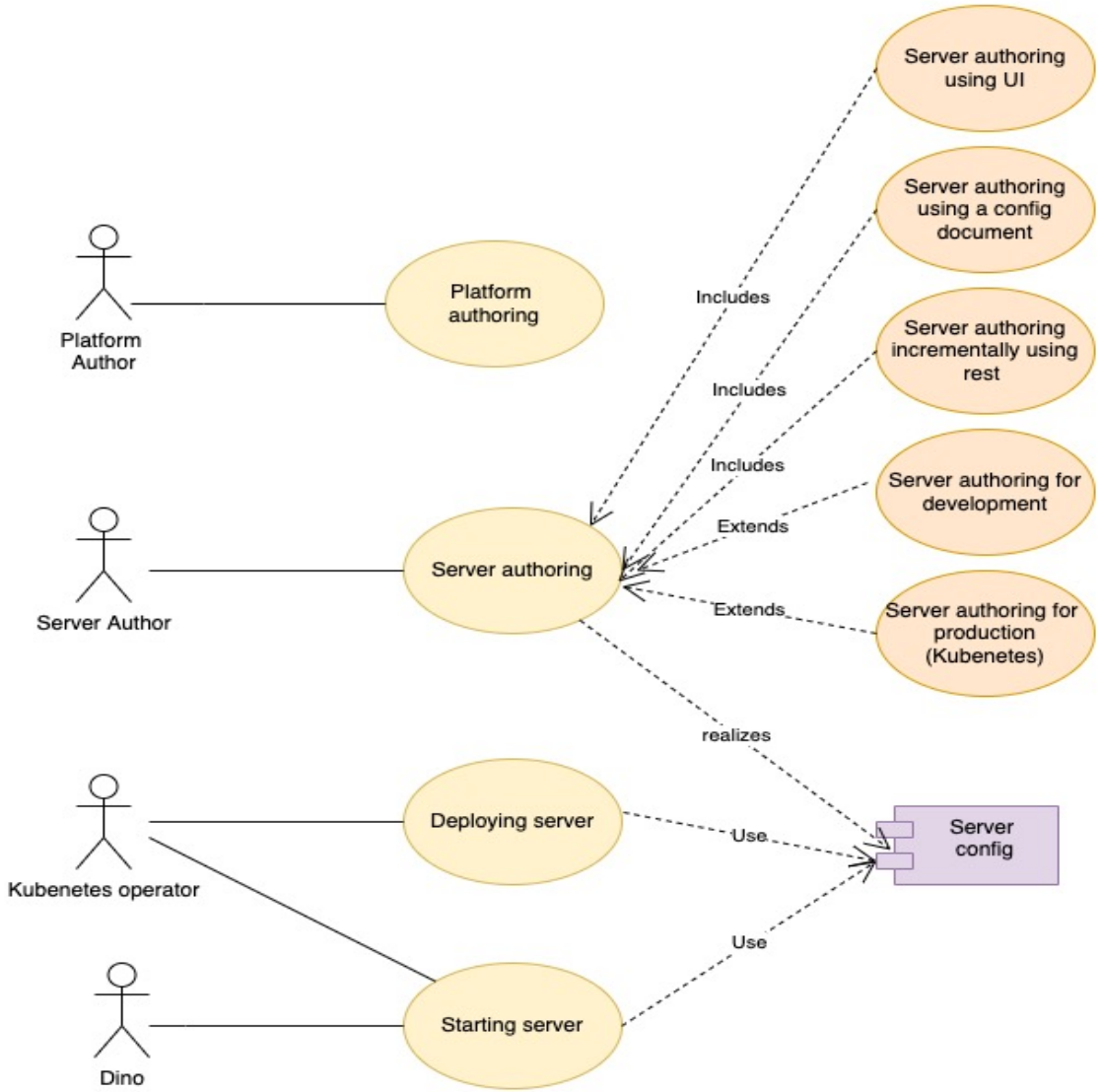
David Radley

Egeria Server Platforms and servers

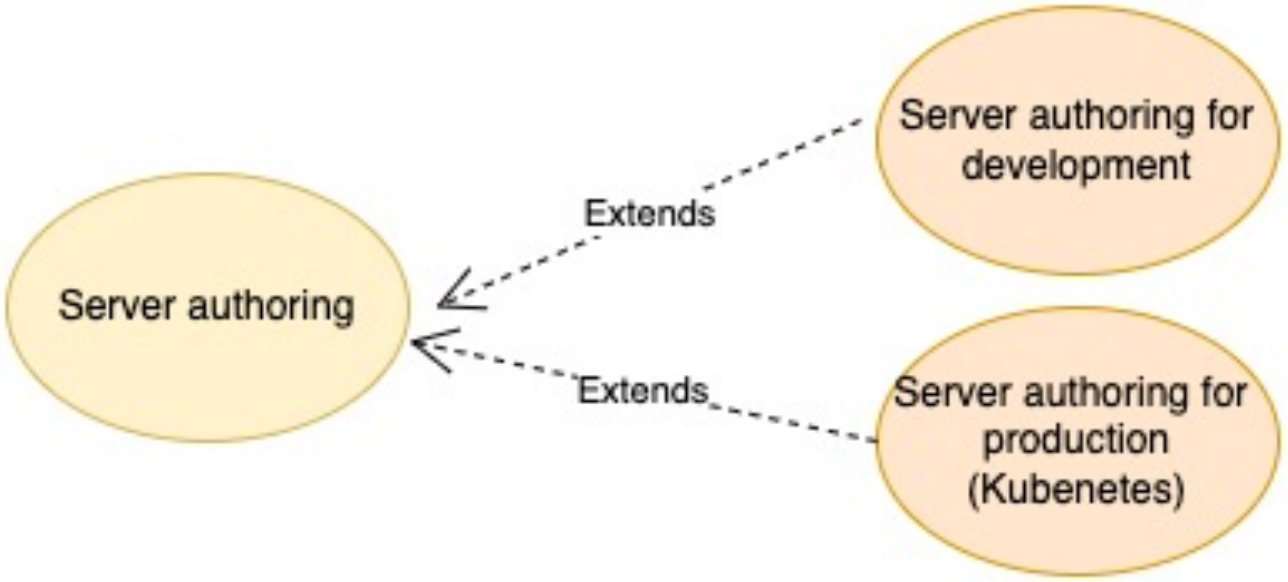
- https://odpi.github.io/egeria-docs/concepts/omag-server-platform/

- Advantages of this architecture.
  - Flexibility to run the same server on many platforms
  - Flexibility to run many servers on one platform.
  - Servers are defined by their configuration (‘metadata driven’); each capability server is not hard coded.
  - There is standard administration calls that are used to configure servers and their capabilities.
  - Framework supports multitenancy

Use cases around server authoring

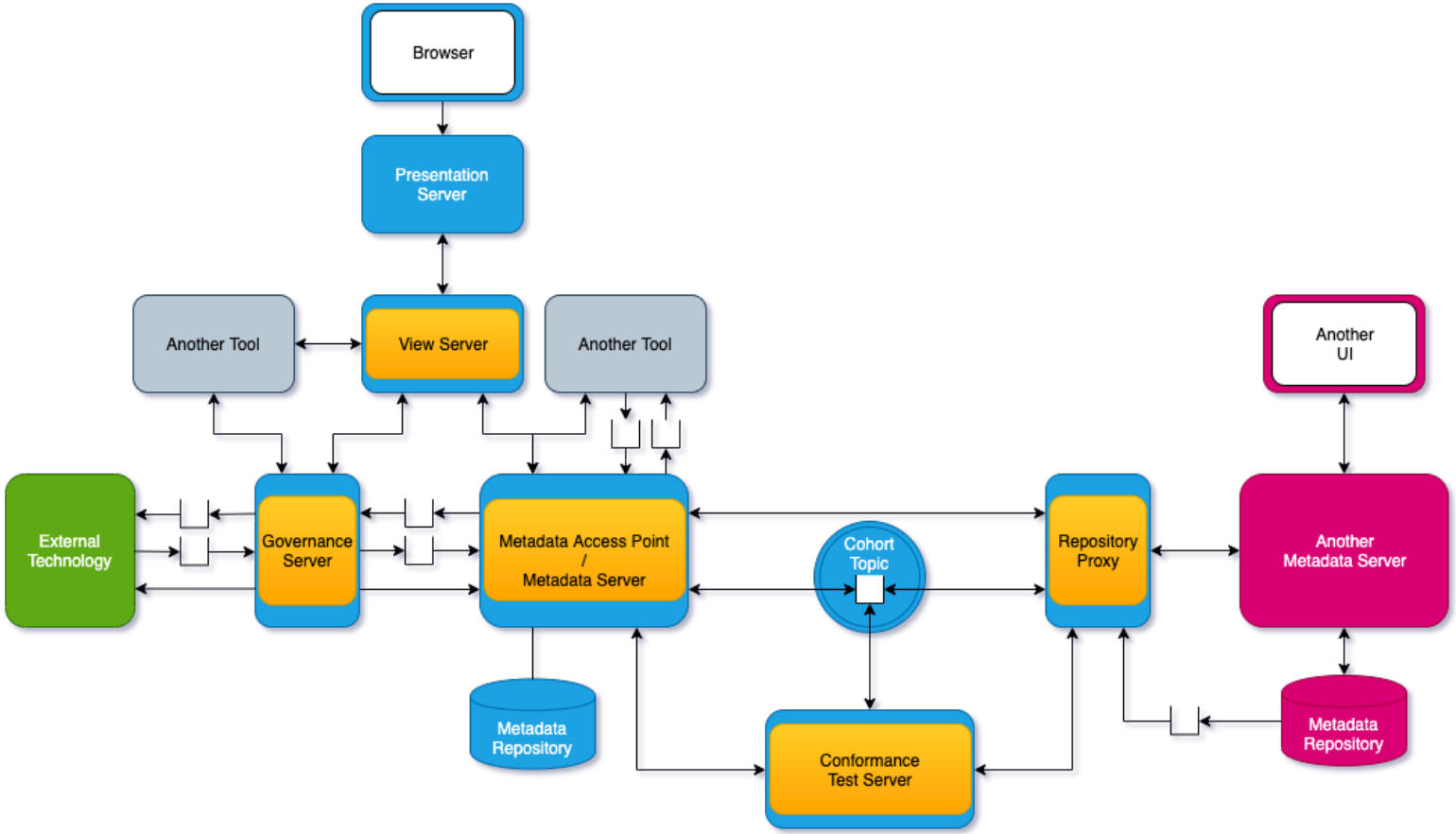


Use cases around server authoring continued



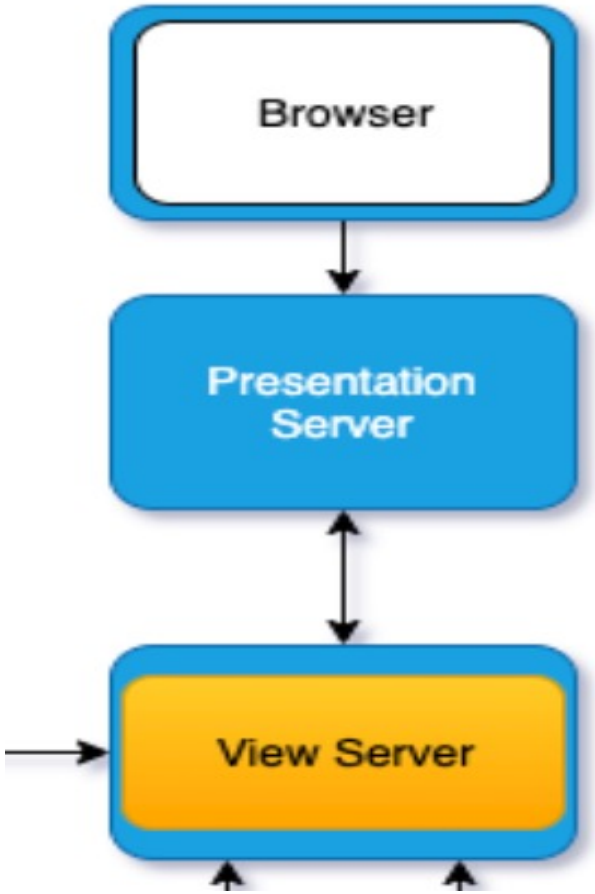
- Development simpler environment
  - often one platform
  - Server run where configured
  - Use rest or Dino to start servers
  - Config Values hard coded
- Production
  - Server configured on a different platform to where it will run
  - Kubenetes operator may be used
    - starts / restarts servers
    - Values need to be templated e.g. URLs cannot be hard coded.

The Egeria infrastructure UI architecture



https://egeria.odpi.org/open-metadata-publication/website/planning-guide/

The UI part

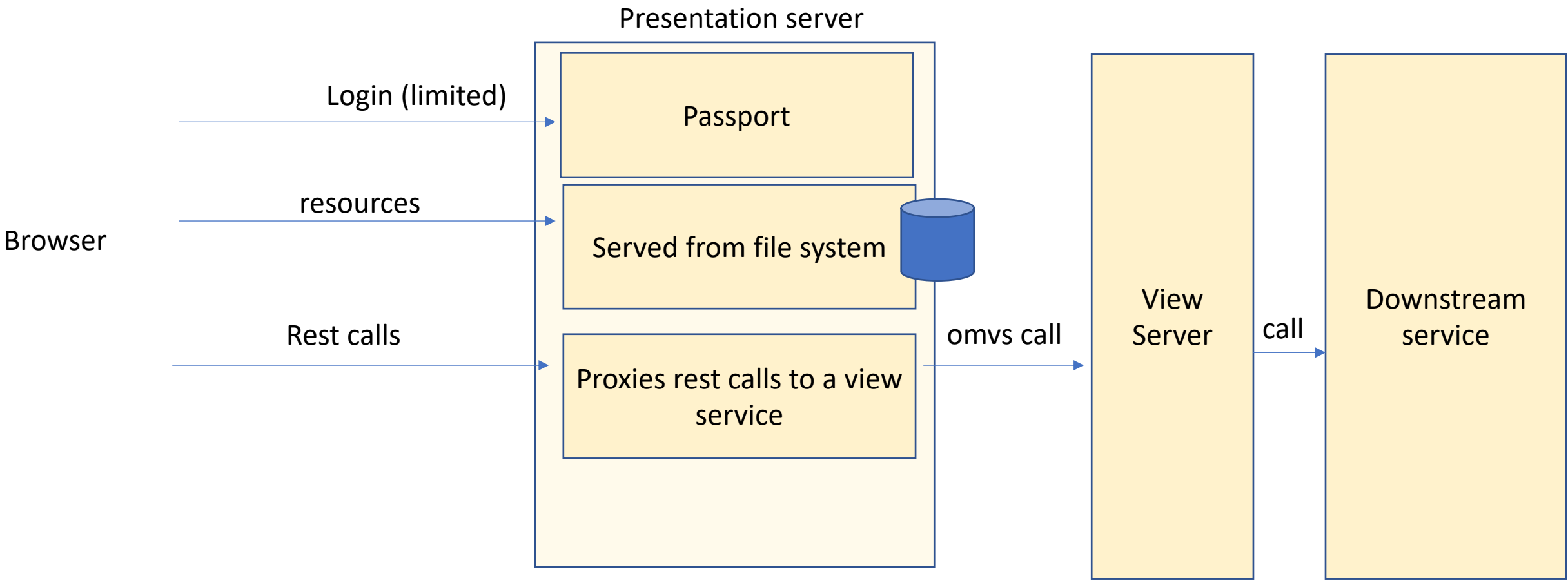


Express based server, uses passport for authentication, webpack for build and client side functional React components with hooks, uses Carbon components. It is tenant aware; tenant information is in environment variables to indicate the View Service endpoint details.

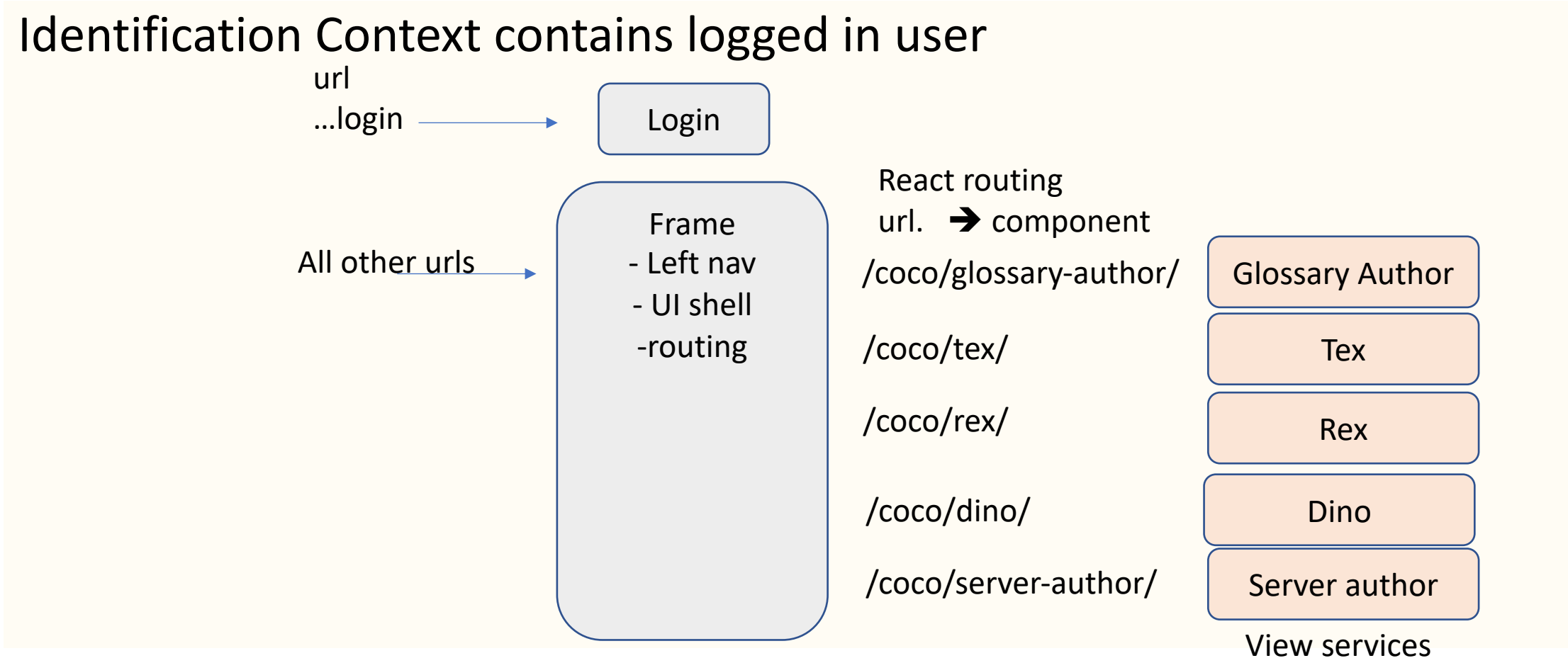
View server is a tenanted view of metadata for UI consumption. It is a type of OMAG Server (so is configured in the usual way)

Demo

React UI architecture – presentation server



React UI architecture – React part



Ui and view services

UI Capability	UI Capability type	View Service	View service target	description
Type Explorer (Tex)	Ecosystem Tool	tex	OMRS	Exploration of the Egeria Type system loaded on server(s)
Resource Explorer (Rex)	Ecosystem Tool	rex	OMRS	Exploration of OMRS entities and relationships present on server(s)
Dynamic Infrastructure and Operations (Dino)	Ecosystem Tool	dino	OMRS	Exploration of the operation landscape
Glossary Author	Solution	Glossary Author	Subject Area OMAS	Glossary author tasks, including Glossary Terms, Categories and their relationships
Server Author	Solution	Server Author	Admin server and platform services	Server author tasks including authoring new server configurations

https://odpi.github.io/egeria-docs/user-interfaces/ecosystem/ecosystem-ui-planning/#ui-capabilities

Server author View service configuration

```
{
"class":"IntegrationViewServiceConfig",
"viewServiceAdminClass":"org.odpi.openmetadata.viewservices.serverauthor.admin.ServerAuthorViewAdmin",
"viewFullServiceName":"ServerAuthor",
"viewServiceOperationalStatus":"ENABLED",
"omagserverPlatformRootURL":"https://localhost:944
"resourceEndpoints" : [
{
"class" : "ResourceEndpointConfig",
"resourceCategory" : "Platform",
"platformName" : "Platform1",
"platformRootURL" : "https://localhost:9443",
"description" : "This platform is running on David's Mac"
}
}
```

Platform endpoint where the configuration will be stored

Platform endpoint definition for platform1

- Note that the view service provides the readable name and description for the platform. And a list of allowable platforms to be used by the server author.

Server author development

- The initial server author Ui wizard was created by a GBS team in IBM. It was hard coded to work for a metadata server (now called access server store).
- Development continued later to add the other server types.
- This involved a lot of refactoring to create some reusable code and patterns that would work for more than one server type.

Server types

- In the React code, there is a json file that defines each server type in terms of its serverConfigElements (each capability is a page in the wizard). Like this:

const serverTypes = [

{

id: "metadata-server",

label: "Metadata Access Store (previously called Metadata Server)",

description: "Supports the access services and supports a metadata repository that can natively store open metadata types as well as specialized metadata APIs for different types of tools (these APIs are called access services).",

serverConfigElements : ["server-type-config-element", "config-basic-config-element", "local-repository-config-element", "esb-config-element", "cohort-config-element", "access-services-config-element", "archives-config-element", "audit-log-config-element", "final-config-element"]

}, ….

https://github.com/odpi/egeria-react-ui/blob/main/cra-client/src/components/ServerAuthor/components/defaults/serverTypes.js

serverConfigElements

A json file defines each serverConfigElements in terms of its identifier , description and label - like this:

const serverConfigElements = [

{

id: "server-type-config-element",

label: "Server Type",

description: "Set the type of server to be configured."

},

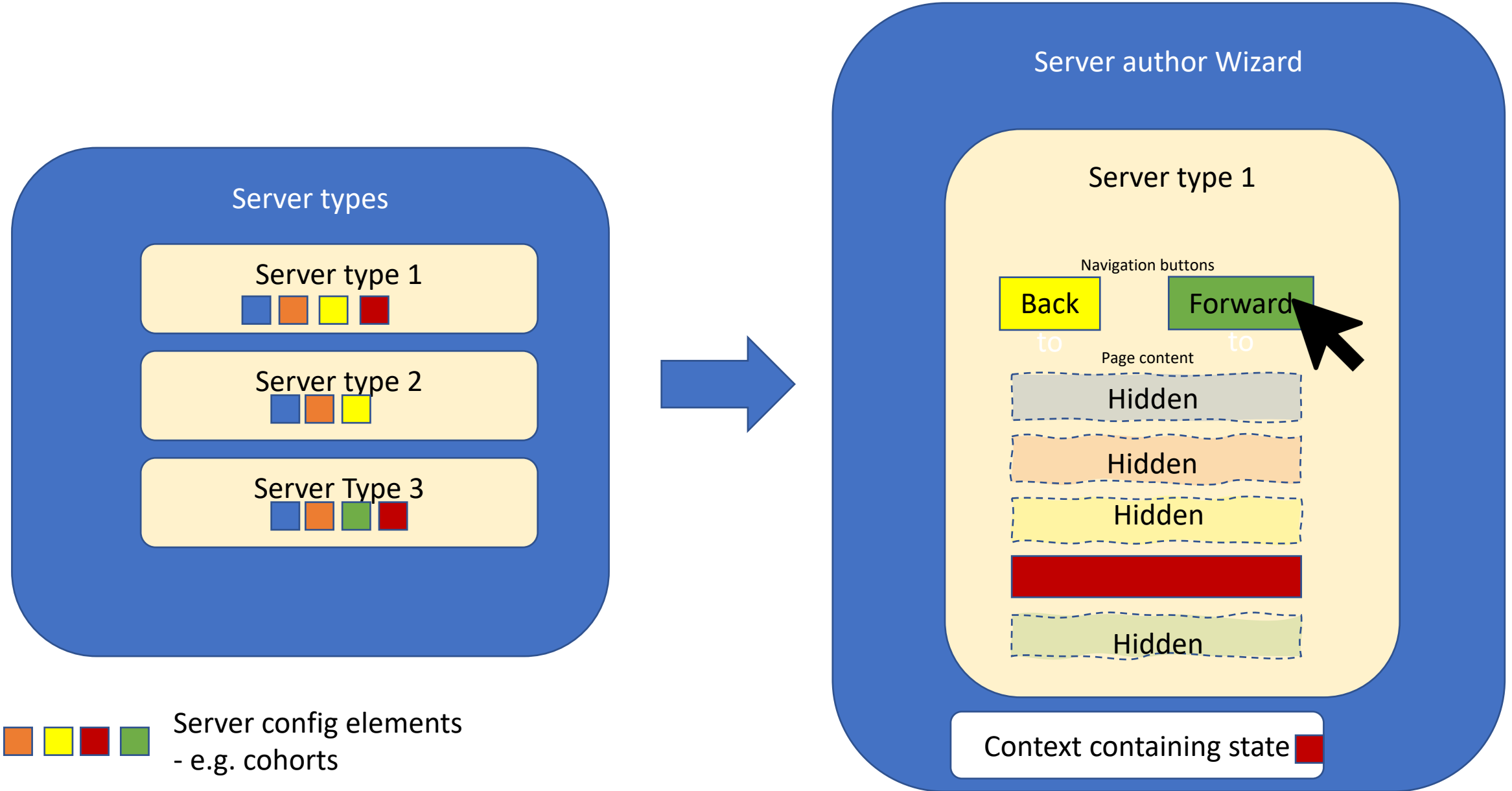
…

https://github.com/odpi/egeria-react-ui/blob/main/cra-client/src/components/ServerAuthor/components/defaults/serverConfigElements.js

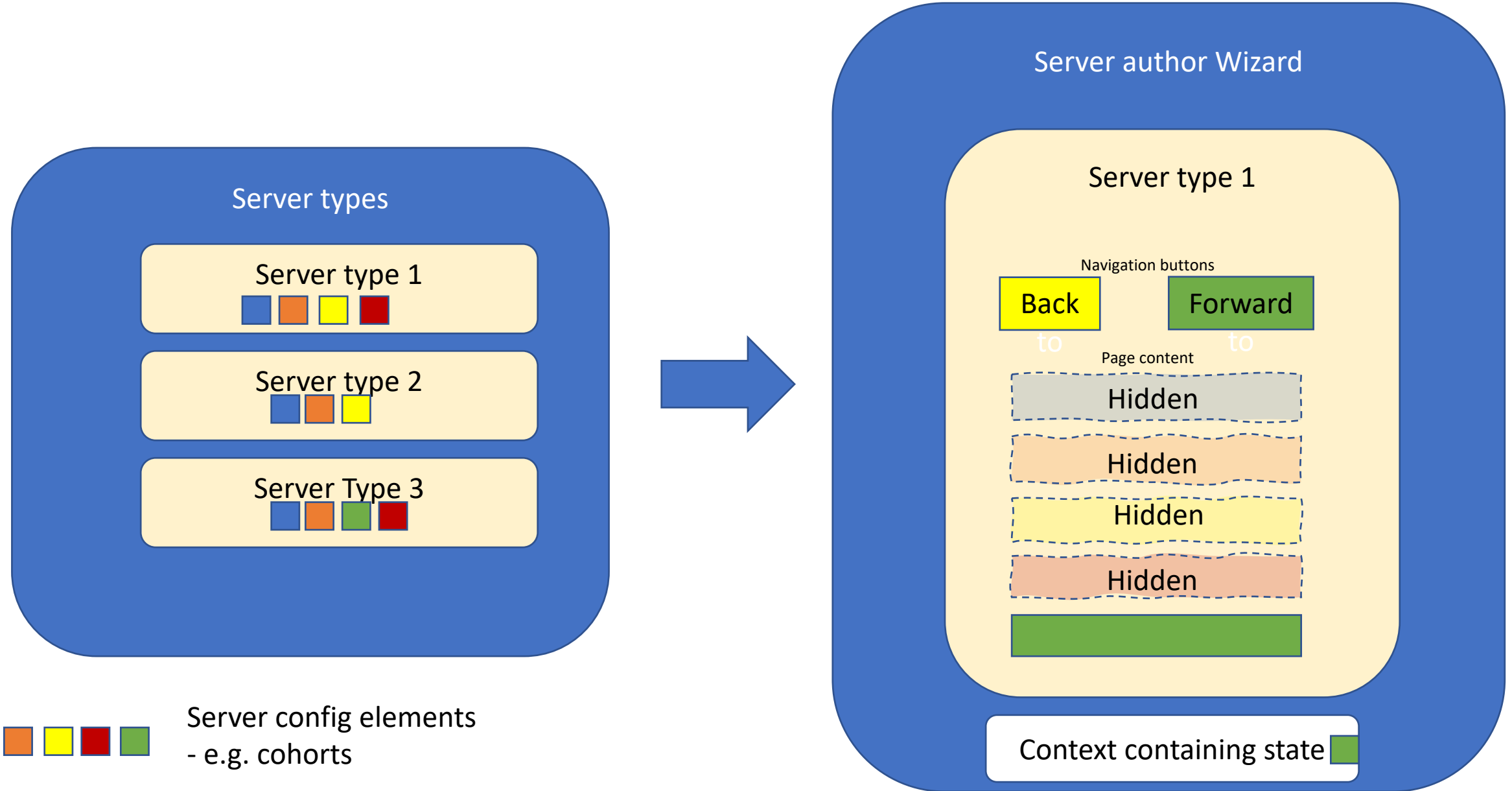
Server author Wizard

- The server author has a wizard to create servers.
- It uses a React context to hold the state.
- The wizard has navigation buttons (back and forward buttons), these interrogate the ServerTypes file and context state to work out what the next step
- Each page is in the wizard component and the appropriate section is unhidden for a wizard page. The ids of the elements match the server config element ids.

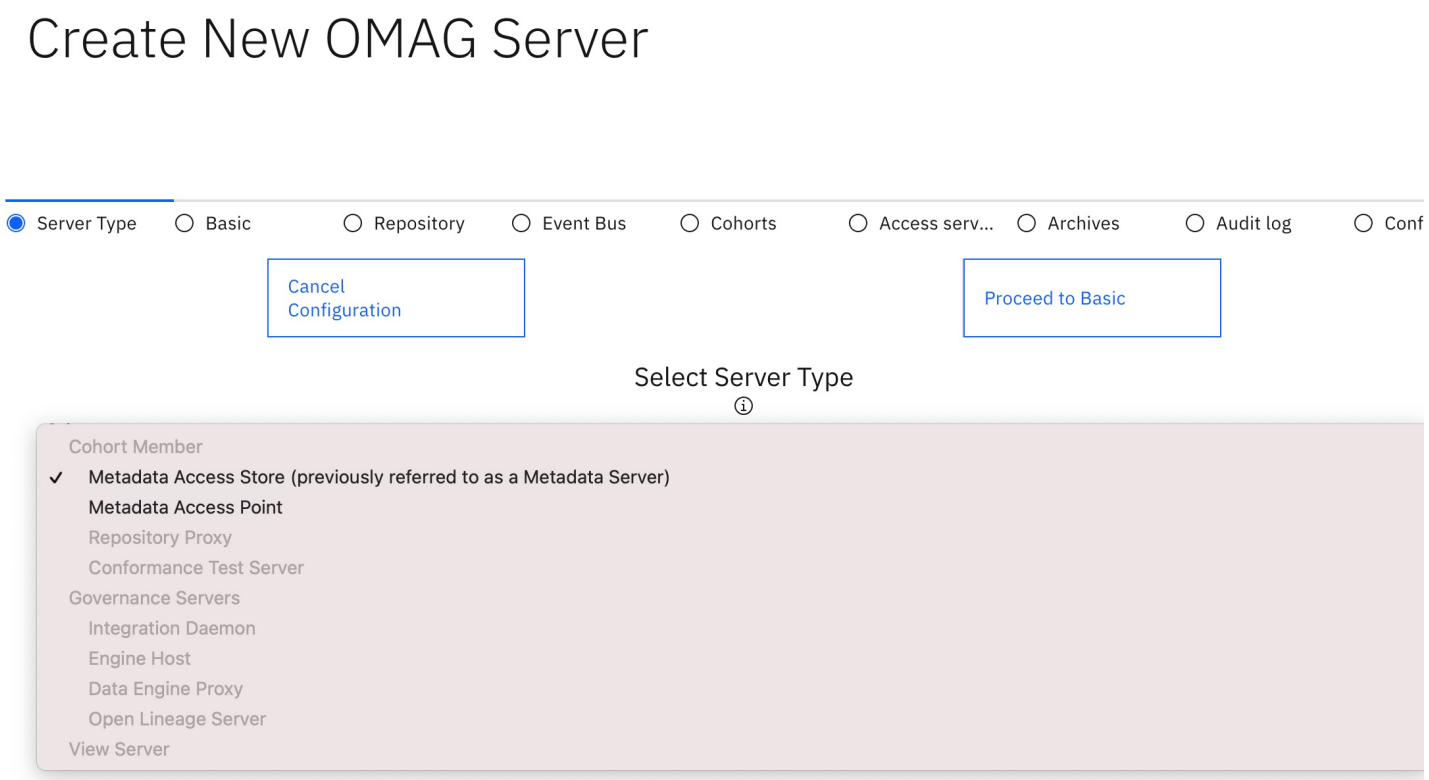
Server types and config elements & context state determines what is shown in the wizard



Server types and config elements & context state determines what is shown in the wizard

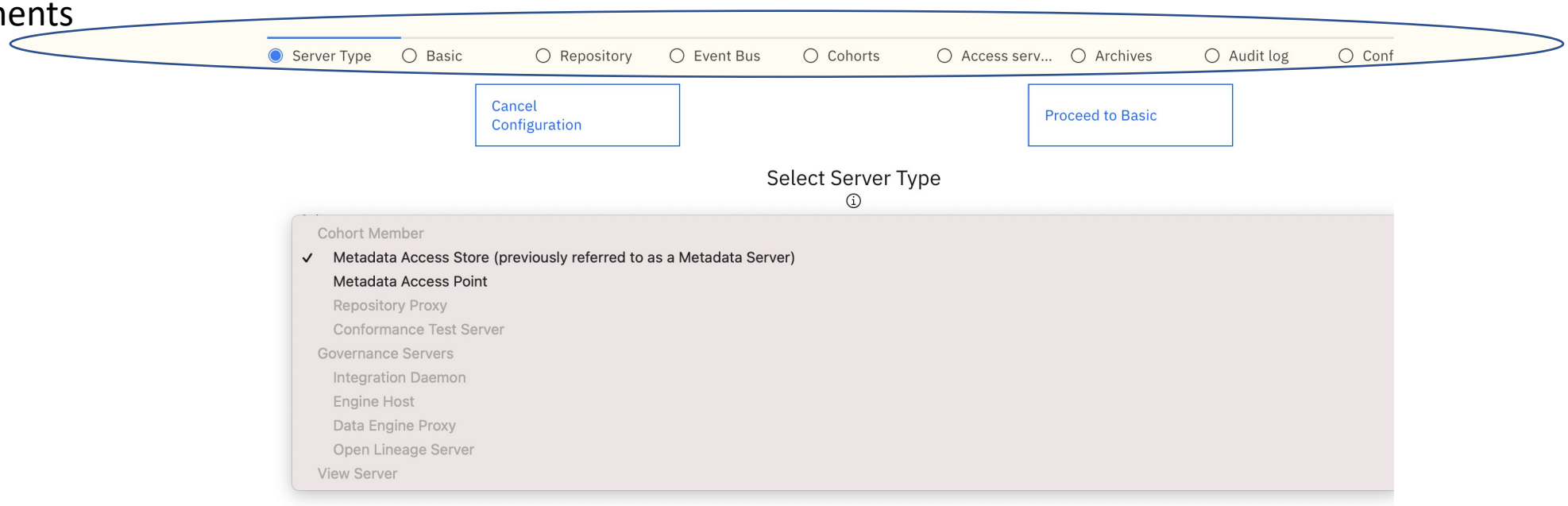


Setting the server type



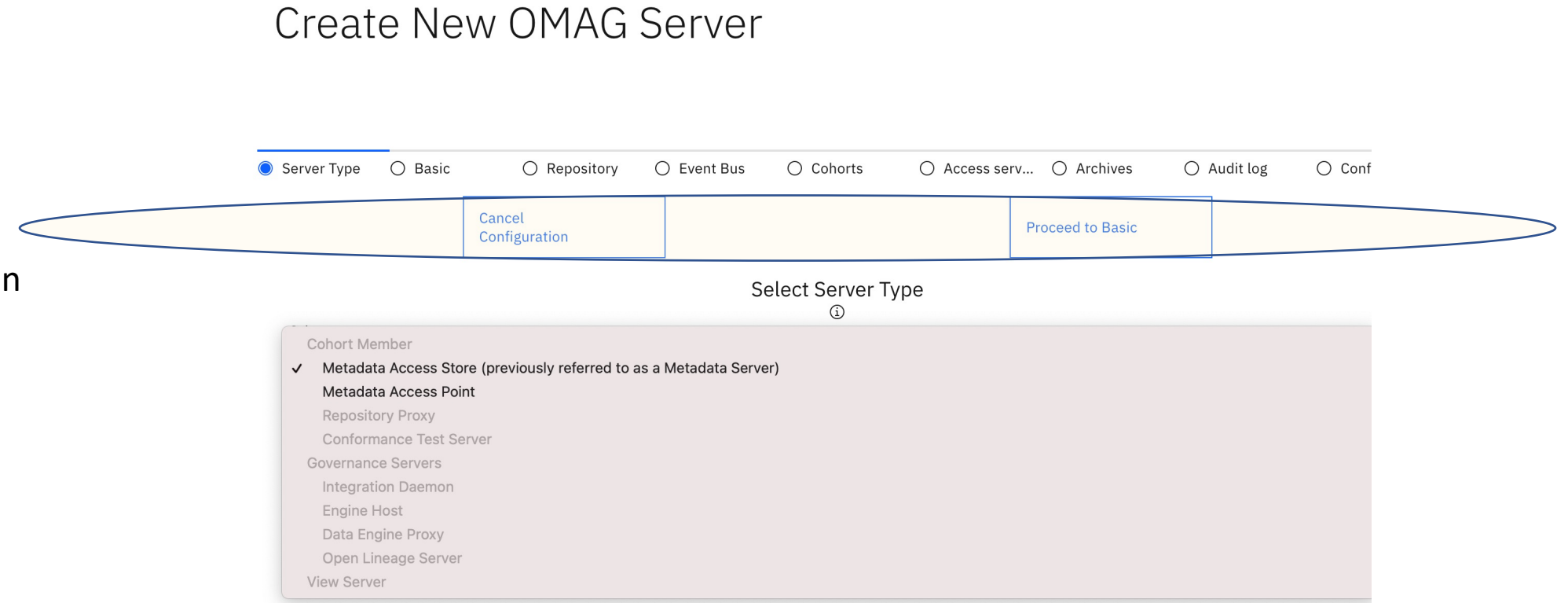
Setting the server type

serverConfigElements



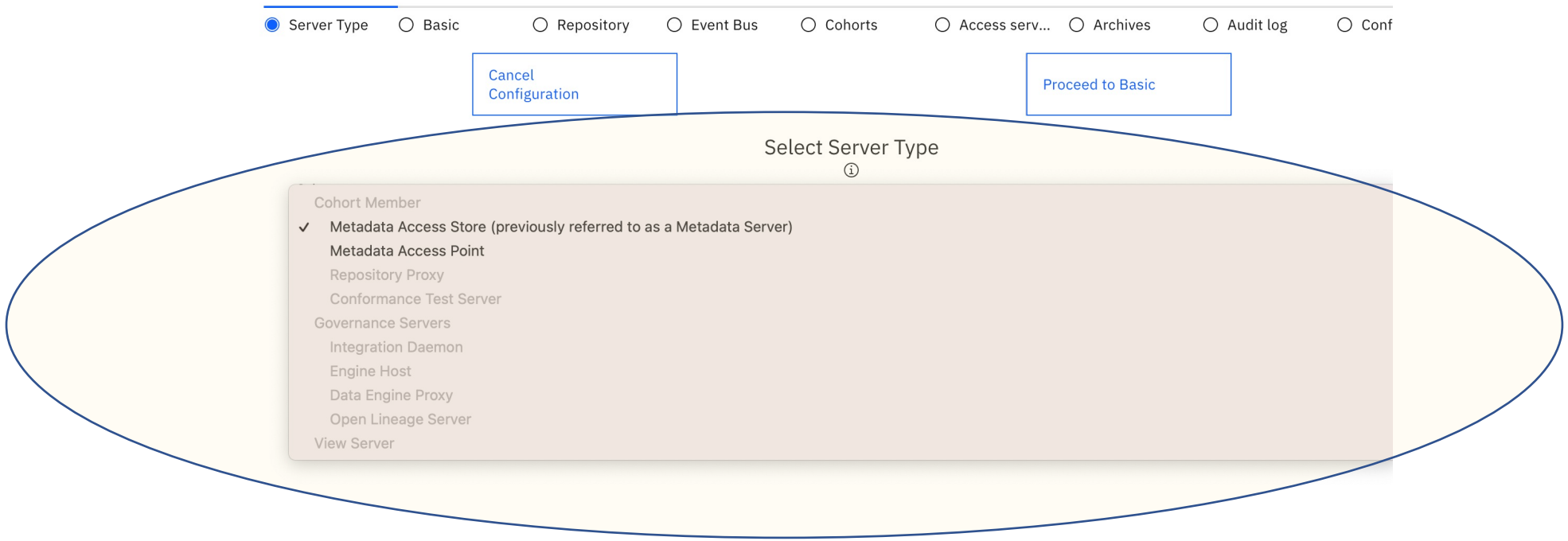
Setting the server type

Server and step specific navigation



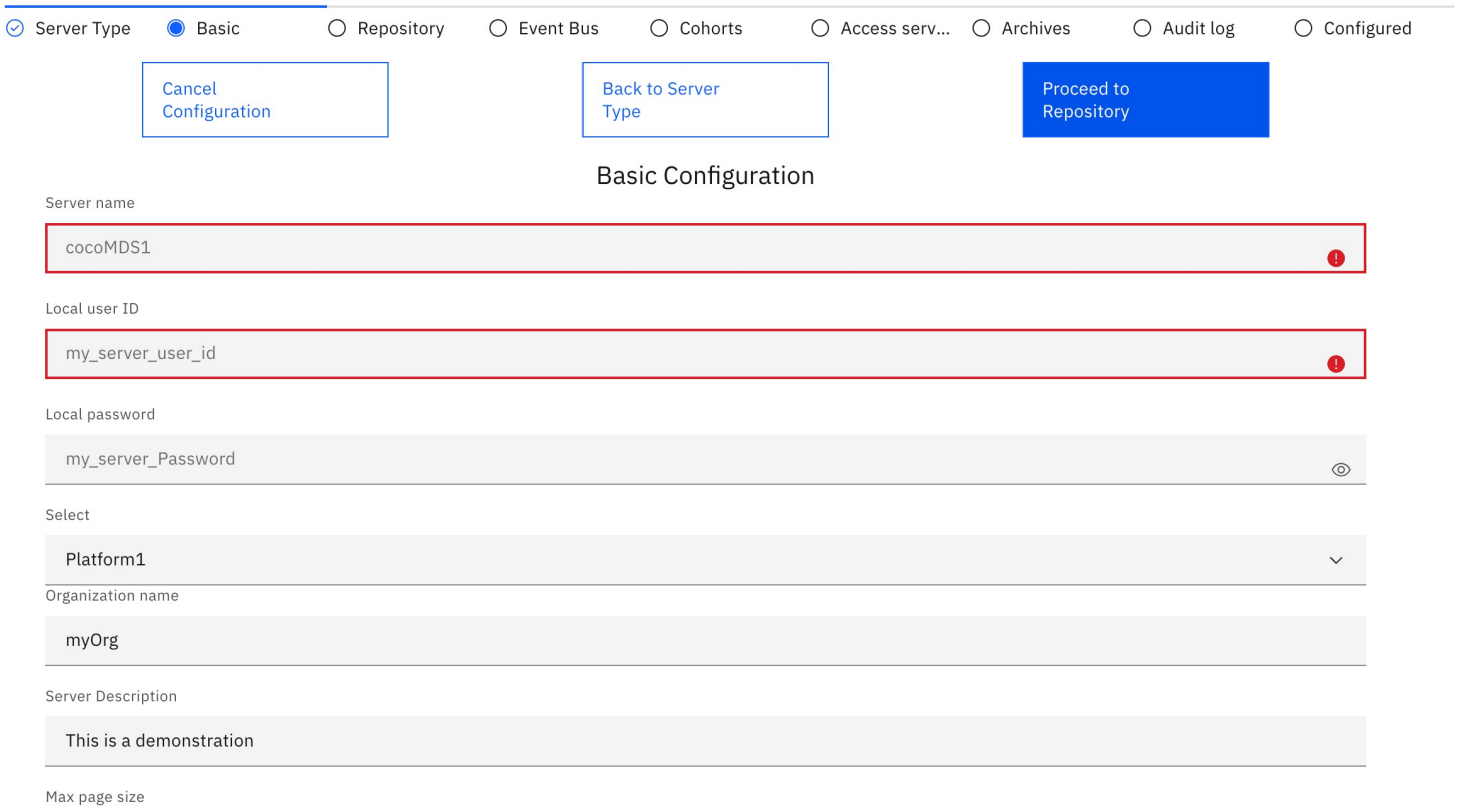
Setting the server type

Unhidden content



Basic config

Create New OMAG Server



Platform
Is the meaningful
Name rather than the
url.

Audit log destinations

- Use a table , with create , edit , copy capabilities to author audit logs.
- This pattern is going to be used in the access server options authoring.
- I am investing React render child pattern to allow re use of component to do this.

Server author current development goals

- Copy, Edit and delete for existing servers
- Implement the other server types
- Bootstrap: From platform(s) with no servers , how do we start?
  - We need a server author view server to be defined and configured.
    - Use admin calls in the presentation server to create this
    - Once we have a view server & server author view service defined we need to start it
    - Then we can start defining the other types of server using the view services as usual
    - consider how we can augment rex tex and glossary author view service config when we are using the server author in that platform.
- Get platform values from a metadata server. For more details see the planned Community call – Egeria Catalog yourself. Coming later this month