Plan for provisioning a Mediaflux collection with portal configuration

This document describes the steps required to provision a Mediaflux collection including an associated portal.

1. Define storage to Mediaflux using the provisioning commands.

    a. Defining the data storage to Mediaflux.

    ```
    asset.store.create :description "RDS_Allocation_Description" :name
    "RDS_Allocation_Name" :path "pathToStorageinFileSystem" :type "file-system"
    ```

    b. Use the Mediaflux provisioning commands to create the collection in Mediaflux, setting up user access.

```
asset.project.create :description "RDS_Allocation_Description" :name
"RDS_Allocation_Name" :namespace "/rdsi" :metadata-namespace "true"
:dictionary-namespace "true" :project-roles < :administrator < :actor -type "user"
"qrisdata:username" > :visitor < :actor -type "user" "qrisdata:username" > >
```

    c. Allow access to the `document:namespace` and `dictionary:namespace`

    ```
    actor.grant :name RDS_Allocation_Name:administrator :type role :perm <
    :resource -type "document:namespace" "RDS_Allocation_Name" :access
    ADMINISTER >

    actor.grant :name RDS_Allocation_Name:administrator :type role :perm <
    :resource -type "dictionary:namespace" "RDS_Allocation_Name" :access
    ADMINISTER >
    ```

2. Load data and metadata into Mediaflux.

    a. Prior to loading data into Mediaflux, there is a need to understand the research data structure. Effective Mediaflux asset management, including Portal functionality, is dependent on extracting and including meaningful metadata. Metadata then allows the formation of valid queries which form the basis of data searches/filtering in the Portal. From past experience this process needs to be performed on a collection-by-collection basis with the researcher. Once the data structure is understood, the process of loading the metadata and data into Mediaflux can be started.

    b. e.g. For CLIMAS data, the collection contains a mixture of files that represent different pieces of information. The metadata existed in the naming of the directory structure and the filenames. A series of scripts were manually constructed to extract metadata from the directory names and file names so that valid metadata describing the data assets could be loaded into Mediaflux.

3. Start configuring the `portal.xml` beginning with the Mediaflux-supplied default.
    a. This is the XML file defining the portal. Now that metadata has been loaded into Mediaflux with the data, define the queries used to search/filter the data in the portal.
        i. The main query statement typically includes all the data from the Mediaflux namespace.

        ```
        <query>namespace&gt;='/rdsi/RDS_Allocation_name</query>
        ```

        ii. Each sub collection defined in a portal contains a query further filtering the data to be presented and a list of filters applicable to the collection. The particular selection of sub-collections and filters depends on the expected audience for the portal. The below XML is an example. This is why the metadata loaded in section 2 is important.

        ```
        <facets>
        <filters label="Format">
                <mode>single</mode>
                <description>The type of region</description>
                <filter label="Interim Biogeographic Regionalisation for
        Australia">

        <query>(xpath(rdsi.climas:region-classification/classification) =
        'IBRA')</query>
                </filter>
                <filter label="State">

        <query>(xpath(rdsi.climas:region-classification/classification) =
        'State')</query>
                </filter>
                <filter label="National Resource Management">

        <query>(xpath(rdsi.climas:region-classification/classification) =
        'NRM')</query>
                </filter>
        </filters>
        </facets>
        <query>namespace&gt;='/rdsi_by_ref/climas/regions'</query>
        ```

    b. Add reference to institutional default layout and branding CSS (and/or custom CSS)

    ```
    <css>../defaults/institution.css</css>
    ```

4. Deploy the `portal.xml` file

```
asset.set :create true :id "path=/www/aportal/portals/THE_PORTAL_NAME" :type
"application/arc-portal" :label "PUBLISHED" :in file:THE_PORTAL_XML
```