COLERIDGE INITIATIVE

ADRF Framework Data Model and DFAdmin

White Paper

Daniel Castellani

Contents

1. Summary

This white paper describes ADRF framework. The framework is built around a data model which encapsulates all the key organizational entities - projects, data and people – and their dependencies. The framework permits the development of data stewardship, collaboration and metadata documentation modules, while at the same time enforcing security constraints associated with these entities. This document presents:

I. the ADRF conceptual framework
II. the data model
III. the Data Facility Administration (DFAdmin) tool.

2. Motivation

Data facilities as the name implies, need to facilitate access to data through an interface that promotes data discovery, use and collaboration. Secure data facilities also need to facilitate ways to ensure that data stewards have an auditable trail of data access and use. These goals are best served at scale by designing and implementing a coherent conceptual framework that addresses both. In the next section we will present that framework, the data model which reflects the structural design, and DFAdmin, which is the operational implementation.

3. Current Approach

3.1. The ADRF Conceptual Framework

The ADRF conceptual framework is user driven and considers the main needs for researchers, data owners and program managers. Federal, state and local data owners need management and secure data stewardship. Analysts and researchers need access, discovery and collaboration. Federal, state and local program managers want analysis and dissemination. At the same time, the ADRF system needs to be secure, reusable, scalable, extensible and interoperable.

Security in the ADRF is built in the conceptual framework from the beginning. The ADRF security is designed to ensure that data security and confidentiality are enforced on five different levels: (i) at the level of safe settings or infrastructure; (ii) at the level of safe people, making sure only authorized users have access to the system; (iii) at the level of safe projects in access-controlled workspaces and services

which ADRF users can make use of in their work; and at the level of (iv) safe incoming data as well as (v) safe data outputs through extensive review by system administrators and data stewards.

Operationally, this means that it is important to design one system that captures the underlying dependencies among people, data and projects. There are several workflows from ingesting data and creating new user accounts, requesting and approving projects and access to data, to formally reviewing research outputs so they can be exported from ADRF. Most of these workflows depend on a variety of information about the 3 main entities (people, data and projects) while each of them require specific and intrinsic details about second-level entities such as data agreements, document signatures (digital or not), trainings, etc. These different workflows and facilities require the flawless integration of several other systems and subsystems. Training personnel to operate all of these system and tools is costly in operations and training. Consequently, another requirement - from the data facility - is to reduce the system complexity by centralizing access and control and automate workflows to reduce manual errors and operational costs. Figure 1 shows an implementation of the conceptual framework using DFAmin, presented in Section 3.3.

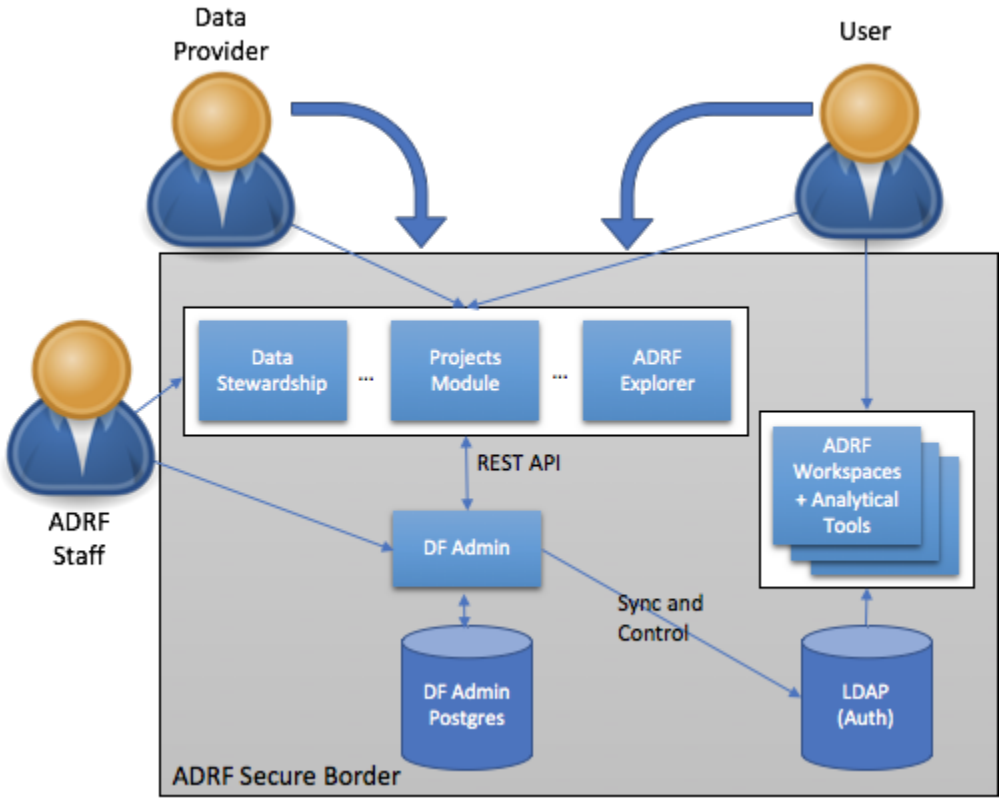

Figure 1. DFAdmin implementing the conceptual framework on ADRF.

3.2. Data Model

The ADRF conceptual framework gains form on the ADRF Data Model, which formalizes the relationships for the primary entities (people, data and projects) and all related secondary entities. The data model was created considering all control, audit and provenance requirements as well as data facility staff and end-user needs. Figure 2 shows a class diagram for the ADRF Data Model.

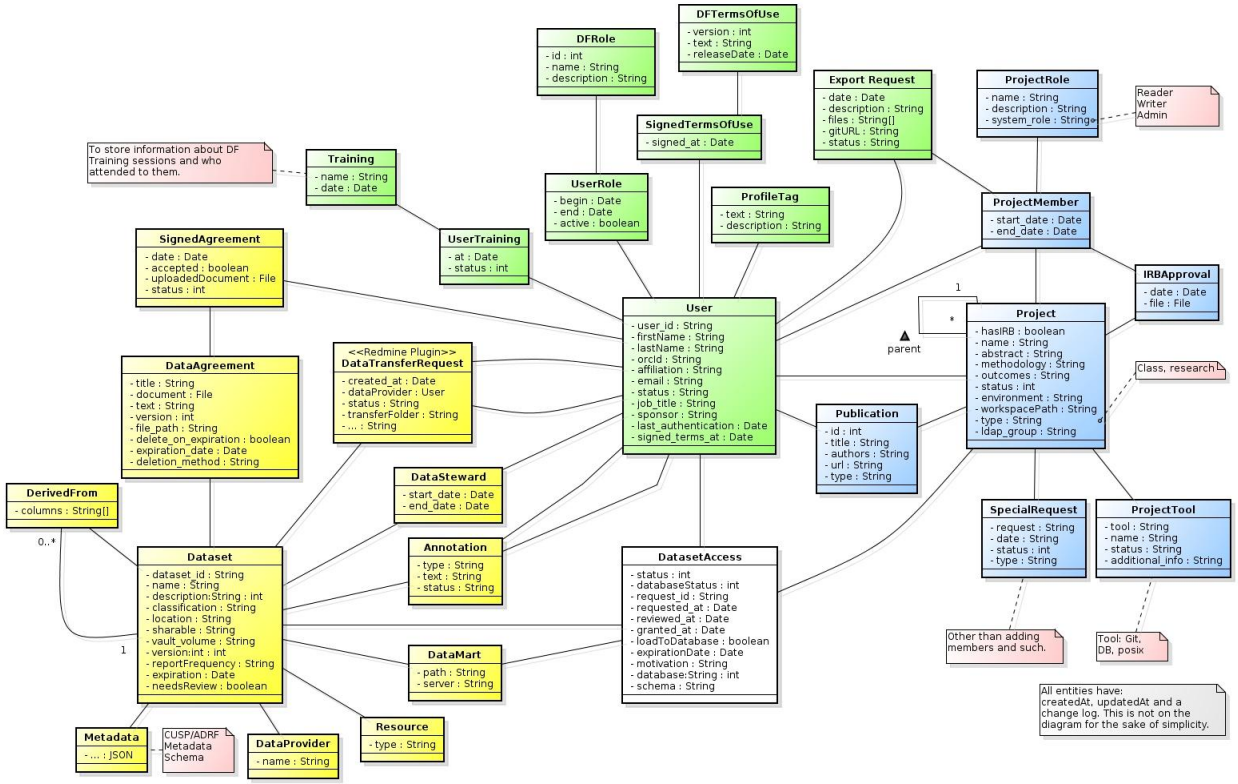

Figure 2. ADRF Data Model.

The entities on yellow are the ones more related to the Data, the ones in green are more related to People and the ones in blue are more related to Projects. The entity in white - *DatasetAccess* - contains information about which people has access to which datasets through which projects. This model is constantly evolving as ADRF evolves and more features are developed and processes are streamlined. In the next paragraphs we present more details about the data model focusing on the 3 primary entities. Historical and audit informational entities were removed from the diagram for the sake of simplicity.

The Dataset entity is the central point of information about data. It contains the information about the data agreements, annotations, data transfer requests, data stewards, etc. With these entities we can have control over the dataset lifecycle and the processes more related to data rely heavily on them. Data Stewards are individuals nominated by the data providers to be responsible for approving access to data. Data Curators, a Data Facility Role (*DFRole* - green, on the top), are staff members with permission to

ingest and curate datasets. The main processes related to data are ingestion, access request/approval, data preparation and archival.

People is another primary entity and ADRF has various information to support its process and day-to-day activities, such as trainings, terms of use and roles. People represent both the end-users (researchers, data providers, instructors or class participants) and staff (data curators, admins, support, etc.) and their respective roles on the system. As a secure system, ADRF need to maintain its users prepared and up to date with trainings such as security and privacy for example. The Training represents this information and stores when the given person took which training. This information enables staff members to know when a new update training is needed or if someone is prepared to have other roles within the organization. Profile tags help to identify different groups of users.

On the ADRF framework, the people's permissions to access data are always related to projects. Therefore, it is crucial to control projects tidily. On the project side, ADRF has information about project membership and which roles the users have, such as member, instructor or student. Different roles can have different sets of system-related permissions (reader, writer, admin). It is also important to know if projects require or have IRB approval and the related publications. Projects can have access to a vast number of tools and it is also important to take this into account to know the project's resources. Some project processes that are based on these entities are project request and evaluation, project creation, project membership management, and project archival, among others.

Of course, ADRF's main purpose is to enable data usage and collaboration. Hence, it has information about who participates in which project and has access to which data. As ADRF is a controlled environment, any data by-products need go through a thorough disclosure review process before they can be exported. This is one of the examples of processes that depend on various entities related to the primary entities (data, people and projects) such as the datasets that a given project has access, who is in that project and which other data they have access and from which project this export is related to.

3.3. Data Facility Admin

The ADRF conceptual framework gains form on the ADRF Data Model and life with the Data Facility Admin (DFAdmin). These tools implement the ADRF Data Model and the Conceptual Framework and is a central point of control to ADRF and all other tools. In this section we will present DFAdmin in more details.

DF Admin is a web accessible administrative portal built inside of the ADRF to manage users, projects, data, access and all ADRF processes. Its technical underpinnings are a Django[1] web application frontend connected to a PostgreSQL[2] database and it is open source. It provides an interface for system

[1] Django is a free and open-source web framework, written in Python, which follows the model-view-template (MVT) architectural pattern. It is maintained by the Django Software Foundation (DSF). www.djangoproject.com

[2] PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance. www.postgresql.org

administrators to see all the datasets in the ADRF, who has access to them, and to modify user and project access to those datasets based on access policies. At any given moment it is possible to know the current and historical status of the Data Facility using the integrated ADRF Data Model presented on the previous section that is implemented on DFAdmin. The ADRF Data Model is also extended with other entities that expand the control information and provide full audit over the admins actions.

DFAdmin reduces the complexity of the system by providing a central interface to manage all the data facility. DFAdmin and a set of scripts control the different sub-systems of ADRF, such as the databases, Gitlab[3], LDAP, Keycloak and many others. From creating a user account to adding a super user to the Postgres database, DFAdmin is the de-facto tool to control the ADRF. For example, when a new user is created, DFAdmin creates this user account and groups on LDAP[4], synchronized Keycloak and set a temporary user password and send a welcome email to the user. Previously, these tasks were manual and error prone, requiring specific skills to know how to manage LDAP and Keycloak[5] - for this single process and several credentials for different services.

Moreover, DFAdmin provides an abstraction layer that facilitates the implementation of applications which enables extensibility and reusability on ADRF. The DFAdmin API has REST endpoints to access the entity related information as well as purpose-related endpoints which provide consolidated information for client-applications. DFAdmin was created with software engineering best practices in mind with the purpose of increasing maintainability, flexibility and extensibility. On Figure 1 it is possible to see some of the ADRF modules that can consume DFAdmin APIs.

Project tools integration

In the ADRF Data Model, each Project Member has a System Role associated that defines the level of access associated with project tool. The system role can be Administrator, Writer, or Reader. The Administrators can add and remove members of the Project. The writers can write in each of the objects mentioned above. Finally, the readers can read in each of the same objects.

When the project information comes straight from DFAdmin, the role is preserved. And it can be reflected in the object. For example, in GitLab the writers have the developer permission and the readers have the reporter permission. (https://docs.gitlab.com/ee/user/permissions.html). On the other hand, if the Project information comes from LDAP because of some limitation, the role cannot be preserved, and a

[3] GitLab is a web-based Git-repository manager with wiki and issue-tracking features, using an open-source license, developed by GitLab Inc. www.gitlab.org

[4] The Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. LDAP is used to provide authentication and group information to several systems inside ADRF. www.openldap.org

[5] Keycloak is an open source software product to allow single sign-on with Identity Management and Access Management aimed at modern applications and services. Keycloak implements the Oauth protocol and provides MFA and OTP to ADRF services, including GitLab and JupyterHub. www.keycloak.org

default role will be applied to all the members of the project. For example, on Linux ACL that is given by groups coming from LDAP, as LDAP doesn't have the role information, inside the project folder the member have write permission.

4. The Future

The ADRF Conceptual Framework, Data Model and DFAdmin have proven to be successful in implementation. However, we expected additional improvements

DF Admin was designed to be integrated with the GMeta metadata schema. Future development of DF Admin will include more granular dataset metadata and editing that will update records as they appear in ADRF Explorer. Additional features to DFAdmin will also include tools for examining and defining relationships between datasets based on similarity between their metadata and specifically the similarity between variables within datasets as well as export of dataset metadata for sharing with other systems.

As ADRF evolves we expect to add support to more automated workflows and applications based on DFAdmin. Automated reports and dashboards for Data Providers showing data usage and results as well as enabling access to other users with less privileges to DFAdmin are also examples of other features being designed. The priorities of these and other features are user-driven as all other aspects of ADRF. Therefore, the ADRF's, and consequently DFAdmin's, evolution is organic and result-oriented.