FERRY: access control and quota management service

Mine Altunay[1], *Joseph* Boyd[1], *Bruno* Coimbra[1], *Kenneth* Herner[1], *Krysia* Jacobs[1], *Farrukh* Kahn[1], *Tanya* Levshina[1,*], *Brian* McKittrick[1], *Rennie* Scott[1], *Timothy* Skirvin[1], *Felix* Stores[1], *Jeny* Teheran[1], *Margaret* Votava[1] and *Tammy* Whited[1]

[1]Fermi National Accelerator Laboratory, Computing Sector, Batavia, IL, USA

Abstract. Fermilab developed the Frontier Experiments RegistRY (FERRY) service that provides a centralized repository for access control and job management attributes such as batch and storage access policies, quotas, batch priorities and NIS attributes for cluster configuration. This paper describes the FERRY architecture, deployment and integration with services that consume the stored information. The Grid community has developed several access control management services over the last decade. Over time, services for Fermilab experiments have required the collection and management of more access control and quota attributes. At the same time, various services used for this purpose, namely VOMS-Admin, GUMS and VULCAN, are being abandoned by the community. FERRY has multiple goals: maintaining a central repository for currently scattered information related to users' attributes, providing a Restful API that allows uniform data retrieval by services, and providing a replacement service for all the abandoned grid services. FERRY is integrated with the ServiceNow (SNOW) ticketing service and uses it as its user interface. In addition to the standard workflows for request approval and task creation, SNOW invokes orchestration that automates access to FERRY API. Our expectation is that FERRY will drastically improve user experience as well as decrease effort required by service administrators.

1 Introduction

The Scientific Computing (SC) Division within the Fermilab Computing Sector provides comprehensive data processing and distributed computing framework for Fermilab's scientific stakeholders. The Frontier Experiments RegistRY (FERRY) service, developed by SC, allows numerous services, including NIS [1], EOS [2], HTCondor [3], dCache [4] and others, to have access to a single source of accurate information related to user identity mapping, authorization attributes and various quotas. Although it is not a core objective of the project, a side benefit is eliminating several obsolete services such as GUMS [5] and VULCAN [6]. VULCAN was an in-house developed service that was used by the Compact Muon Solenoid (CMS) collaboration at the LHC Physics Center (LPC) at Fermilab that

[*] Corresponding author: tlevshin@fnal.gov

provided fine-grained user mapping to specific resources. It also allowed us to retire the VOMS-ADMIN [7] tool which provided the user interface to the VOMS [8] service.

The Fermilab Human Resource Database (user_db) contains information on current and retired employees, visitors and summer students. Only active Fermilab users in good standing have a Fermilab Services account that provides access to Fermilab email, SharePoint, and ticketing services. Only a subset of these users are people who need to run jobs on the Grid and transfer data to and from Fermilab storage services. Such users are usually affiliated with one or more Fermilab experiments and projects (see Figure 1). Access to interactive and Grid clusters, storage services and storage quotas, batch priority and slot allocation depends on their affiliation to a particular experiment. Prior to development and deployment of FERRY this information had been stored by various services and was often out of sync. FERRY created a single source of truth for this data.

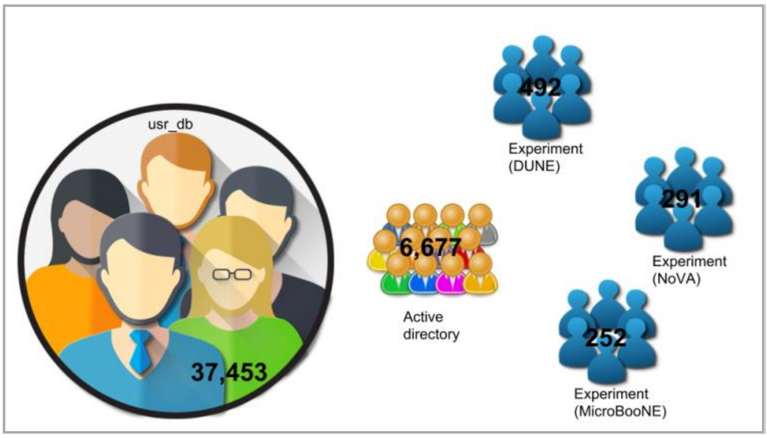


Fig. 1. Groups of users that have been registered in FERRY, from the Fermilab HR database (user_db) to VO membership.

2 Requirements

Some of the essential FERRY service requirements, that came from multiple stakeholders and customers, are listed below. The FERRY service should:

- Provide flexible APIs that allow services to pull information about:
    - VO role and grid map-files
    - VO members, their certificates, groups and roles
    - LPC users, their CERN attributes, EOS quota and LPC group affiliations
    - Linux passwd and group files for a specified VO and/or compute resource
- Be able to extend and modify data schema. The schema should support:
    - users with multiple VO memberships and groups
    - resource-oriented groups that control access to a resource (NIS passwd and group files, storage access, … etc.)
- Be available during local working hours. The dependent services should rely on cached information.
- Provide appropriate level of security for data access. Though none of the information stored by FERRY is considered sensitive, it still gathers a lot of information related to users in one central place.

- Integrate with ServiceNow (SNOW) [9] to handle customer requests. Failure to propagate data from a customer request submitted via SNOW should be periodically retried if the FERRY service is down and a ticket to the FERRY support group should be opened if the retry limit is reached.
- Allow for removing a user and all related attributes and affiliations.

3 Architecture

The FERRY service consists of a RESTful Web Service and a FERRY database. Custom SNOW forms have been created to fill specific requests by invoking the corresponding API call to the RESTful Web Service. The SNOW service uses the "Orchestration" [10] technique to communicate with FERRY. Service providers' scripts are querying the FERRY service using more than eighty APIs to get relevant information. Only a small subset of clients are authorized to modify FERRY data via FERRY APIs. Clients use x509 certificates [11] to authenticate to the FERRY Service. All clients are required to provide a valid x509 certificate and access is only allowed from the local network. In order to make any modification in FERRY, a client's certificate must be whitelisted in the FERRY configuration, or the API call must originate from a trusted local IP address. In the future we are planning to introduce fine-grained authorization at the API call level.

To initially populate the FERRY database, several scripts pulled data from various sources, then correlated and validated them before inserting the values into the database. The update user script, run as a cron job, acquires information about new and retired users from Fermilab's HR database and Active Directory Service and then updates FERRY. All other modification requests come via SNOW. Fermilab services periodically query FERRY to get needed data. The FERRY architecture is shown in Figure 2.

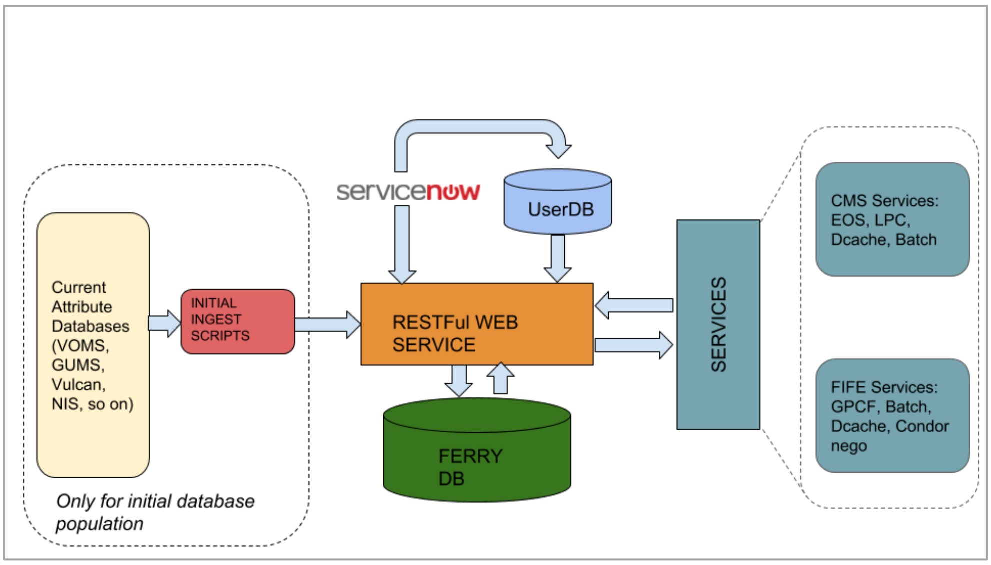

servicenow
UserDB
Current Attribute Databases (VOMS, GUMS, Vulcan, NIS, so on)
INITIAL INGEST SCRIPTS
RESTFul WEB SERVICE
SERVICES
CMS Services: EOS, LPC, Dcache, Batch
FIFE Services: GPCF, Batch, Dcache, Condor nego
FERRY DB
Only for initial database population

Fig. 2. FERRY Architecture.

4 Implementation

The FERRY web service is implemented in the GO language. Postgres was chosen for the database implementation. All FERRY APIs return information in JSON format. The databases tables are logically organized in groups:

- User-related tables that contain a user's full name, linux id, linux name, and user's groups.
- Affiliation unit (Virtual Organization), Fully Qualified Attribute Name (FQAN), unit members and certificates.
- Compute resource and access to it, that include user's primary and secondary groups, preferred shell and home directory.
- Storage resources, access to storage and quotas.

5 Integration with Services

The FERRY service replaces existing authorization services such as GUMS and VULCAN. The information obtained from FERRY, allow the services to correctly authorize users and provide valid account mappings. FERRY enables us to get rid of stale and inaccurate data previously stored locally by various services.

Many grid services need a grid map-file that maps a distinguished name from a presented certificate to a local Linux account. This Linux account would be used to execute a job or access a file on a Linux node. Other services need a VO-role map-file for mapping a Fully Qualified Attribute Name to a local Linux account. The services that need these files pull them from FERRY by using the relevant FERRY APIs.

FERRY is keeping track of members' affiliation with experiments. In the past, there were multiple "sources of truth" for this sort of information; for example NIS tables for experiment interactive clusters, HTCondor CE head nodes and worker nodes all got information from different sources. FERRY also serves as a single source of truth for populating VOMS instances. FERRY contains various other service configuration metadata which previously was stored only in configuration files of individual services. Since these data are now in FERRY, they can easily be modified by SNOW processes, initiated from user request forms, and the services can pull the configuration from the FERRY database at predetermined intervals to configure the underlying services. The information currently stored in FERRY includes HTCondor group account quotas and EOS storage quotas.

5.1 FERRY and Grid services

Fermilab provides computing and storage resources to multiple experiments. These resources are accessed through a gateway known as a 'compute element' or CE. We use HTCondor-CE as our gatekeeper of choice. Part of the responsibility of a CE service is determining if an experiment or a user is authorized to access the compute resources attached to it. HTCondor-CE at Fermilab relies on VO role and grid map-files to make this decision. FERRY, being the source of truth for access control, provides this information to the CE through RESTful API calls. A python script runs hourly on a caching server to fetch these map-files from FERRY and places them onto a local webserver. The CE service then periodically pulls in these map-files and uses them to make authorization decisions. We have ten CEs with different access control requirements. FERRY provides the necessary flexibility to enforce these access control policies per cluster.

5.2 FERRY and NIS

In order to tie FERRY to existing NIS services we use a caching server. This caching server pulls data from FERRY on an hourly basis; validates the data; converts the data into "standard" data formats; and places the output onto a local web server. Each client can then query the data as necessary for its operations.

One example: user and group management replaces NIS with nss_db by using a JSON formatting compatible with json-passwd [12]. Each individual client (worker node, interactive machine, etc) independently pulls down group/passwd data for a specific resource from the FERRY caching server on an hourly basis, and saves it locally in a format compatible with the libnss_db Linux library.

The major benefits of this architecture are:

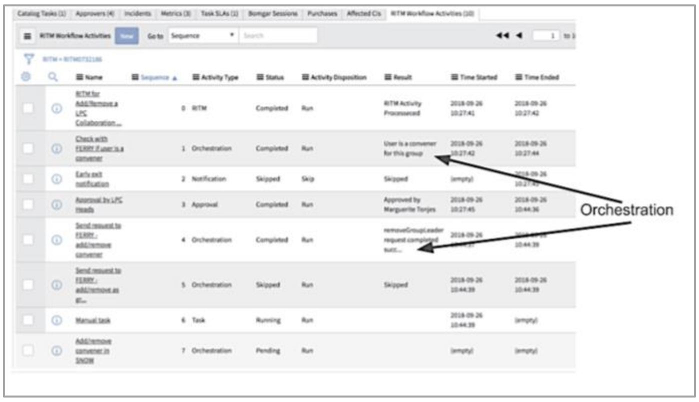
- Simplicity - everything works through simple web calls and basic bash scripting.
- Performance - easily scales to thousands of systems.
- Reliability - clients can continue to get data during FERRY system outages and off-hours, and could even use their existing cached data if network access failed.

5.3 FERRY and VOMS

The synchronization script pulls relevant information from FERRY - such as groups and roles, members and their certificates, as well as member groups and role affiliations - and inserts it directly into the VOMS database. Users with inactive (expired) Fermilab accounts are deleted from the VOMS service automatically.

6 Integration with SNOW

FERRY integration with SNOW provides an improved user experience by allowing the creation of comprehensive request forms with drop down menus populated with valid data choices from FERRY. We have streamlined the request approval/rejection process for people in the approval chain. The major breakthrough was achieved by implementing a push request to FERRY by using the SNOW Orchestration mechanism that is shown in Figure 3.



Name	Sequence ▲	Activity Type	Status	Activity Disposition	Result	Time Started	Time Ended
RITM for Add/Remove a LPC Collaboration...	0	RITM	Completed	Run	RITM Activity Processed	2018-09-26 10:27:41	2018-09-26 10:27:42
Check with FERRY if user is a convener	1	Orchestration	Completed	Run	User is a convener for this group	2018-09-26 10:27:42	2018-09-26 10:27:44
Early exit notification	2	Notification	Skipped	Skip	Skipped	(empty)	2018-09-26 10:27:4[illegible]
Approval by LPC Heads	3	Approval	Completed	Run	Approved by Marguerite Tonjes	2018-09-26 10:27:45	2018-09-26 10:44:36
Send request to FERRY : add/remove convener	4	Orchestration	Completed	Run	removeGroupLeader request completed succ...	2018-09-26 [illegible]	[illegible] 10:44:39
Send request to FERRY : add/remove as gl...	5	Orchestration	Skipped	Run	Skipped	2018-09-26 10:44:39	2018-09-26 10:44:39
Manual task	6	Task	Running	Run		2018-09-26 10:44:39	(empty)
Add/remove convener in SNOW	7	Orchestration	Pending	Run		(empty)	(empty)

Fig.3. Example of SNOW form workflow with orchestration.

This workflow will allow the automation of user registration and attribute management. For example, if a user submits a SNOW request to become a member of a particular experiment, a SNOW form allows them to select an experiment and Grid role within the experiment (e.g Production). The SNOW workflow requests approval from the experiment's coordinator and pushes this information to FERRY once approved. Various services that create passwd/group files on interactive and grid nodes or update VOMS and dCache information will periodically pull this information from FERRY and change the corresponding files and configuration.

7 Deployment

The database used by the FERRY service is deployed in the centrally managed Postgres database cluster maintained by the Core Computing Division at Fermilab. There are production and development instances with nightly backups for disaster recovery. The databases are deployed on a redundant cluster to ensure high availability.

The RESTful API for FERRY is implemented in the GO language. A GIT repo has everything needed to deploy the GO-based RESTful API. The steps involved are:

- Execute Git clone
- Populate a configuration file with user/password for the db
- Execute docker-compose up

These actions allow us to deploy the FERRY service that responds on a predetermined port and talks to the configured database. The docker-compose config file specifies how to build the GO code, copies in the config files it needs, runs the GO app, and also starts up a container that monitors the database, pushing stats to our Graphite service, and another container that updates the FERRY database from central Fermilab databases.

8 Monitoring

We are monitoring both components of the FERRY system and push the data to Fermilab's monitoring infrastructure, Landscape [13]. These data are available through the Open Source visualization tools, like Grafana [14] and Kibana [15]. We are using a docker image named wrouesnel/postgres_exporter to monitor database activity and report it to a Graphite/Grafana installation used by a large number of other services and users at Fermilab. This is a generic tool which pushes metrics from a Postgres installation and allows us to plot many different metrics relating to database activity so we can detect anomalies and understand usage. The database dashboard is shown in Figure 4.

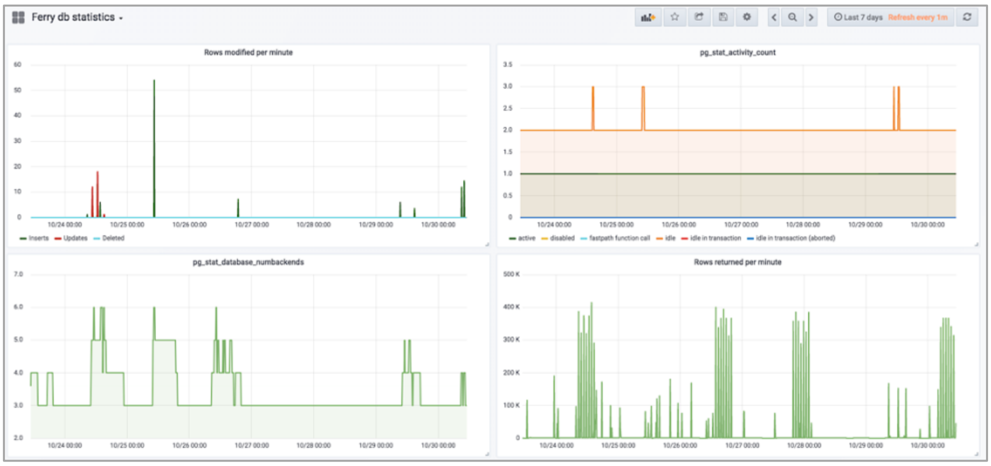


Fig.4. Monitoring database performance using Grafana/Graphite service.

To understand the usage patterns of the individual API calls we are sending FERRY service logs to an ElasticSearch [15] cluster which is also used by many other services and users at Fermilab. The logs of the application are split up into key value pairs which allow us to make plots of anything from individual API invocation counts to plots showing the client locations from where the API was accessed, as well as query content, duration and status. Figure 5 shows an example of a plot that could be created based on log information.

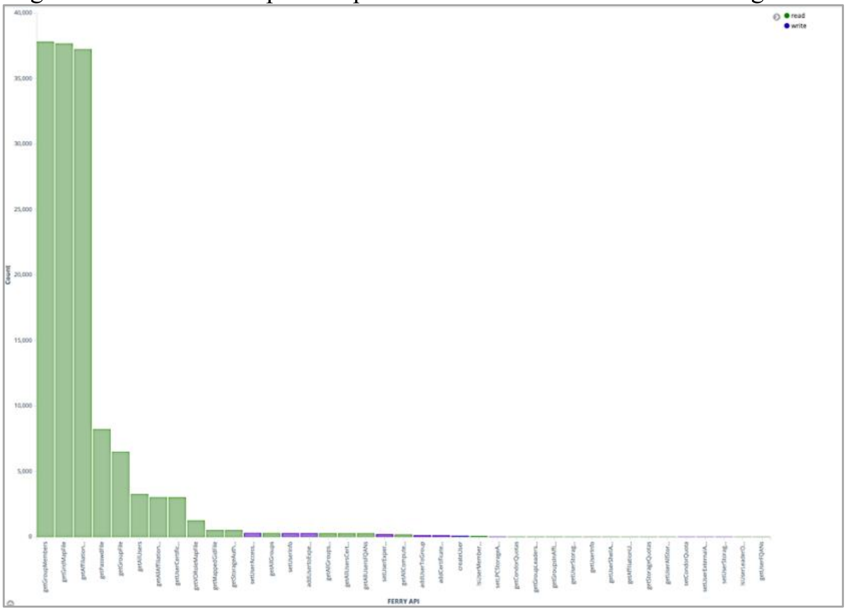


Fig. 5. Weekly graph of FERRY activities by API requests using Kibana/ES.

9 Conclusion

The FERRY service provides a centralized repository for the access control and job management attributes such as batch and storage access policies, quotas, batch priorities and NIS attributes for cluster configuration. It maintains a central repository for previously scattered, and sometimes obsolete, information related to users' attributes. FERRY provides RESTful APIs that allow uniform data retrieval by numerous services. It enables the use of custom forms from ServiceNow and allows orchestration, so users can request services/changes which will be quickly deployed to services in an automated fashion.

With FERRY's deployment we are able to retire unsupported services, such as GUMS and Vulcan, and change the service level support from a 24x7 basis to an 8x5 basis, reducing the total cost of ownership for the organization. Overall, FERRY reduces operational load on support staff while also improving end users' experience.

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non- exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

1. NIS: https://en.wikipedia.org/wiki/Network_Information_Service
2. EOS: http://eos-docs.web.cern.ch/eos-docs/
3. HTCondor Overview: http://opensciencegrid.org/docs/compute-element/htcondor-ce-overview/
4. dCache: https://www.dcache.org/manuals/Book-5.0/
5. GUMS Retirement: http://opensciencegrid.org/technology/policy/gums-retire
6. VULCAN: https://cmsweb.fnal.gov/bin/view/Software/Vulcanindex
7. VOMS-ADMIN Retirement: http://opensciencegrid.org/technology/policy/voms-admin-retire/
8. VOMS: http://repository.egi.eu/2012/07/10/voms-2-0-8/
9. ServiceNow (SNOW): https://www.servicenow.com/
10. SNOW Orchestration: https://www.servicenow.com/products/orchestration.html
11. X509 certificate
12. https://github.com/tskirvin/json-passwd
13. Landscape: https://indico.cern.ch/event/721026/contributions/2964078/attachments/1629322/2615771/Landscape_CERN_201804.pdf
14. Grafana/Graphite: http://docs.grafana.org/features/datasources/graphite/
15. ELK Stack: https://www.elastic.co/elk-stack