

# Landscape Program Requirements and Architecture

*Joe Boyd, Tanya Levshina, Kevin Retzke*

Version History			
Version	Date	Author(s)	Change Summary
1.1	03-21-2016	Tanya Levshina, Joe Boyd, Kevin Retzke	Draft initial document
1.2	05-11-2016	Tanya Levshina	Version update d based on Ruth Pordes' feedback.
1.3	05-13-2016	Tanya Levshina, Joe Boyd, Kevin Retzke	New revision

[1. Overview](#)

[2. Purpose](#)

[3. BackGround](#)

[4. Scope](#)

[5. Objectives](#)

[6. Stakeholders](#)

[7. Customers](#)

[8. Requirements](#)

[8.1. General requirements:](#)

[8.2. Stakeholders requirements:](#)

[8.3. Customer requirements:](#)

[8.3.1. Operational and Service Management requirements:](#)

[9. Landscape Architecture](#)

[9.1. Architectural Diagram for User and Log Analytics Service for FIFE Experiments](#)

- [9.2. Architectural Diagram for GRACC Service](#)
- [9.3. Service and Site Availability Service](#)
- [9.4. Program Execution Architecture](#)
- [10. Appendix A: Justification for Open Source Software Selection](#)
  - [10.1. ELK in HEP](#)
- [11. Appendix B: Landscape and Other SCD Projects](#)

## 1. Overview

The Landscape program's goal is to provide a comprehensive framework to monitor Grid services health, resource utilization, jobs and data movement at HEP Cloud Facility.

## 2. Purpose

The purpose of this document is to

- Capture the requirements of experiments and projects, service providers, user support and management to monitor, analyze, troubleshoot and audit our batch and data handling infrastructure.
- Describe the architecture for the proposed solution.

## 3. BackGround

### 1. Why are we doing this program now?

During the last several years the experiments and management expressed their concern and dissatisfaction with the state of monitoring of Grid jobs. One needed to be aware of multiple services and login to a specific webpage to be able to understand the status and usage of a particular service. All the monitoring pages look differently and one need to be very familiar with the service in order to interpret the content of the page.

More and more experiments are starting to use common FIFE infrastructure and tools. With addition of Public Clouds and increased usage of OSG resources we need to provide a central place for experiments to be able to monitor the status of their jobs and be able to troubleshoot the problems. The Landscape program is designed to do just that.

### 2. What will happen if we do it later?

If we do not start to work on these services as soon as possible we will need to drastically increase the efforts for user support.

### 3. What if we do not do it at all?

There is a chance that we will lose our customers who will try to find dedicated clusters and give up HEP Cloud and OSG usage.

4. Who will benefit from this project?

FIFE experiments, SCD management, and HEP Cloud service providers.

5. Do the people who will benefit from it consider it the most important improvement that can possibly be made at this time?

It would definitely be a big improvement for our customers. It makes them and the service providers enabling their computing usage more efficient by having an in-depth view of the system at all times. Especially, the program will help people responsible for experiments' off-line production.

## 4. Scope

The requirements are limited to those of the FIFE customers. The scope covers

- user jobs submitted to the HEP Cloud Facility,
- health monitoring and alarming of batch and data handling middleware,
- facilitating troubleshooting and analytics including resource consumption and utilization efficiency.

The requirements and architecture will be used to guide the

- selection of mature, stable Open Source products,
- appropriate development, integration and operation of services,
- personnel processes and interactions for service and user support,
- testing and validation activities.

## 5. Objectives

The outcomes of this program are to deliver into production solutions that meet the requirements using a modern, extensible, flexible, scalable design and implementation that is easy to use and enhance as well as adapt and evolve to new needs, tools and capabilities.

The deliverables include:

- Framework into which diverse timed and structured widely distributed system parameters and real-time probes (including user defined) can deliver their information.
- Scalable, robust, federated repository of the information gathered.
- Well specified program and user interfaces for input of and access to the information
- Easily configurable and very user friendly graphical interfaces, dashboards, presentation interfaces for system and user selected collections of the information stored.
- Easy to use presentation options for information to be included in presentations and papers.

- Alert and alarm capabilities based on analysis of the real-time information.
- Support for troubleshooting and system efficiency activities based on system (and user?) specified interpretation of system and user selected parts of the information.

## 6. Stakeholders

We have identified the following groups as stakeholders for this program:

- Users Executive Committee (UEC) representing the users of our distributed computing services and facilities.
- Scientific computing program (experiments and projects) leaders, liaisons, and management
  - Understand resource utilization by their users.
  - Obtain graphical information in standard formats for effective communication to collaborators, overseers and sponsors.
- Computing management and coordinators
  - Plan for resource needs
  - Understand science operations and workflows
  - Prioritize and allocate resources (including people)
- Computing Service Owners
  - Provision of resources for future needs
  - Identify users that are not using services efficiently and provide help.
- The Open Science Grid
  - Understand utilization of OSG resources
  - Report use of the distributed facility to funding agencies,
  - Meet MOU agreements with the LHC experiments by providing information to the WLCG.
  - Meet agreements with the NSF XD program by providing information to the XSEDE accounting services.

## 7. Customers

The following groups will be using Landscape Program:

- **Projects (Geant4, Noble R&D etc.) and experiments (NOvA, Mu2e, DUNE, MicroBooNE, etc.)** that are using the Grid and Cloud infrastructures to monitor and report on the overall throughput and effectiveness of the use of their resource allocations for simulation, data processing and analysis.

- **Users of any distributed resources** at Fermilab will be able to monitor and debug in real time the status of jobs submitted to the Grid, queues and transfer rates for their data, current storage usage, efficiency and success rate of their jobs.
- **Grid and Cloud services support and operations staff** (USDC, OPOS, GSO) will be able to see and receive alerts on the overall status of the services, current HEP Cloud utilization, availability and efficiency of OSG sites for FIFE experiments, status and performance of data management services such as SAM and FTS.
- **SCD management** will also use the service to monitor resource utilization trends and understand provisioning needs.
- **The OSG** (VO admins and users, PIs, sited administrators, OSG ET etc) will use the service to monitor current sites usage, efficiency and success rate.

## 8. Requirements

These are the current requirements collected during our requirements gathering stage. It is, clearly, possible that Landscape dashboards usage by numerous users will generate new requirements and these will be folded into the planning of annual meetings.

### 8.1. General requirements:

- Provide services that allow stakeholders and customers to monitor jobs submitted on HEP Cloud and all other grid and cloud resources at Fermilab, observe data transfers, and troubleshoot Grid and Cloud related services.
- Build extendable, scalable and sustainable services.
- Minimize development by using well accepted open-source software.
- Provide a pluggable architecture where the software modules can be easily replaced.
- Initial focus on incorporating new data sources and new dashboards.
- Allow rapid development and iteration of tailored views for each target audience.
- Preserve historical data accumulated before start of the program.
- Preserve, where possible, existing probes and agents collecting data.
- Provide means to extract data and upload it to any future service.
- Provide reasonable replication of existing data in order to have backup in case of data lost.
- Minimize manual work to add new experiments and users.
- Services should be easily reinstalled and configuration management tools should be used to facilitate services installation.
- Provide means to add additional metrics and analytics.
- Provide means to modify/correct historical information.

## 8.2. Stakeholders requirements:

- Monitor in real time (per user/experiment)
  - Submission rate, idle queue
  - Utilization, efficiency of HEP Cloud Resources (FNAL HTCondor and AWS)
  - Utilization of the OSG sites
  - Data transfer rate and size for various services including dCache, Enstore, gridFTP, xrootd, hadoop etc
  - dCache queue, SAM and FTS statuses
  - dCache tape usage
  - Current priority and allocation of resources
- Have means to generate monthly/yearly plots based on historical information (per experiment/role)
  - Wall Hours, CPU spent on jobs per facility
  - # of User Jobs
  - Efficiency and Success Rate
  - Wall Hours, CPU spent on glidein pilots on GPGrid
  - Pilot job efficiency
  - Data transfer rate and size
  - Storage Usage
  - Tape usage and transfer size
  - Number of users using facility
  - Charges accumulated by using Public Clouds
- Collaborate with other groups that are involved development of monitoring and accounting for Grid infrastructures, including OSG, XSEDE and WLCG:
  - Share toolkits wherever possible
  - Outreach and communicate to CERN, OSG, XSEDE and other relevant organizations.

## 8.3. Customer requirements:

Scientific User (project, experiment etc):

- Have means to requests dashboards tailored for experiments.
- Monitor in real time:
  - Status of jobs
  - Availability of resources
  - Requested vs utilized cpu, memory, duration
  - Pre-emption rate on OSG sites
  - dCache, FTS, SAM queues and status
  - Transfer rates
  - dCache Pool usage

- Ability to troubleshoot:
  - The reasons for jobs being idle
  - The reasons for jobs being on hold
- Have means to generate monthly/yearly plots based on historical information (per experiment/role/user) (same as for stakeholders)
- Access to the running job detail:
  - Where it is running (site, host)
  - What resources it is using

#### Service Administrators:

- Monitor in real time:
  - HepCloud Resource usage
  - Status of HEP Cloud services (condor, Factory, Frontend, jobsub)
  - Priority and efficiency
  - Public Cloud usage, charges and spot prices

#### Other Service Providers:

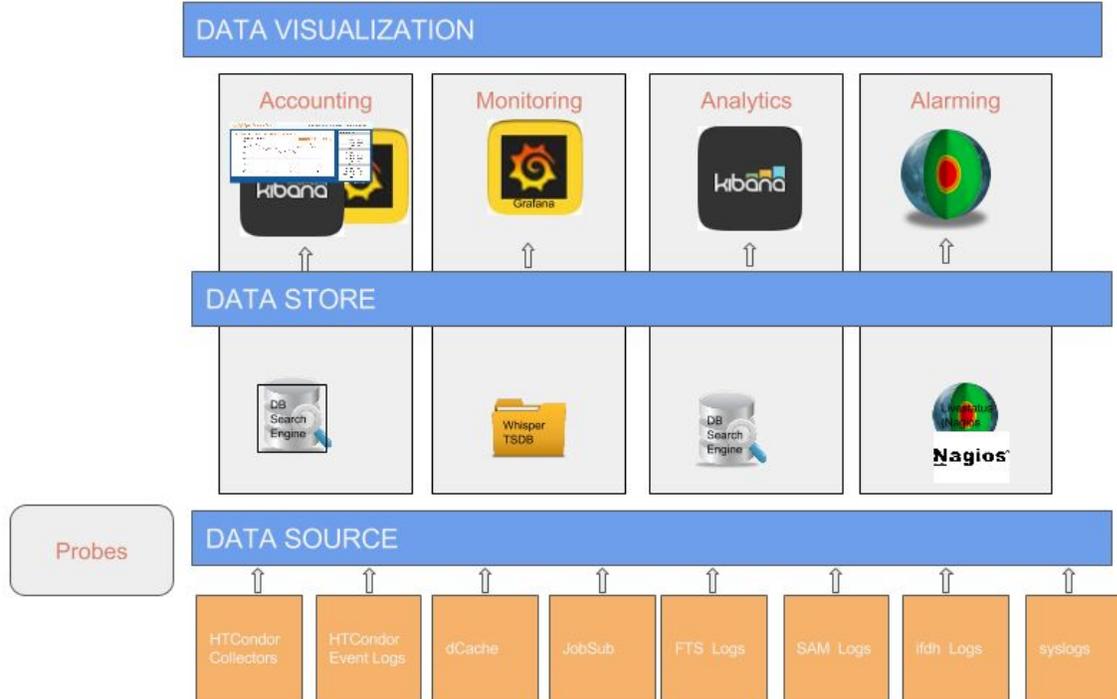
### 8.3.1. Operational and Service Management requirements:

- Services availability is 95% for Job Monitoring and Accounting services. Service and Site Availability Service should be available 99.9%.
- Services support is limited during working hours (5 days 9 by 5)
- Most of the Landscape views will be accessible for anyone with a Services login. Single Sign On will be implemented.
- Some of the monitoring dashboards will be protected and be available for a limited set of people (e.g access to pricing information for Public Cloud).
- Operation should perform data replication/backup. Data should be archived and be possible to restore data from the archive.
- Software needs to be upgraded according to release schedule.
- Some operational work needs to be performed on demand, e.g experiment name is changed.
- Services availability should be monitored by the existing alarm service (e.g check\_mk) and alarm should be sent to operation team in case of service unavailability. The appropriate checks need to be activated to monitor disk, cpu and memory usage on the machine where services are running.

## 9. Landscape Architecture

We distinguish several layers in the program architecture: Data Collection services, NoSQL databases for data storage, and visualization platforms with support for numerous data sources and authorization methods. Components used in each layer will be replaceable, encapsulated and independent.

The high level architecture diagram is presented on the figure below:



### 9.0.1 Data Collection Layer

Data collection probes will be represented by numerous custom scripts and third party agents (e.g Logstash client) capable of collecting and forwarding data to the Data Store layers. We will reuse with minimal modification the probes that have been developed during previous years. Some of the existing probes are listed below:

- Gratia probes (include collection of job records, data transfers, storage usage, etc)
- Generic HTCondor probe
- SAM and FTS probes
- dCache queue probe
- Blue arc locks probe

Probes will be running on remote nodes, collect information and forward it to Data Store. New probes could be added at anytime.

### 9.0.2 Data Store Layer

We propose to use NoSQL Open Source products, such as Graphite and Elasticsearch, as a data store. Both tools have a query facility that can be optimized for time series data. Graphite is well suited to deal with a large number of unique, independently distributed statistics at discrete, bucketed time intervals. The stats should be conceptually hierarchical: they have either parent-child or sibling relationships with each other. It contains one or more archives, each with

a specific data resolution and retention policy for long-term retention of historical data. So, the data we are collecting from HTCondor such as number of running, idle, held jobs, memory and cpu usage per job, transfer rate to dcache, number of files declared at SAM or queued at FTS's dropbox is well suited for storing in Graphite.

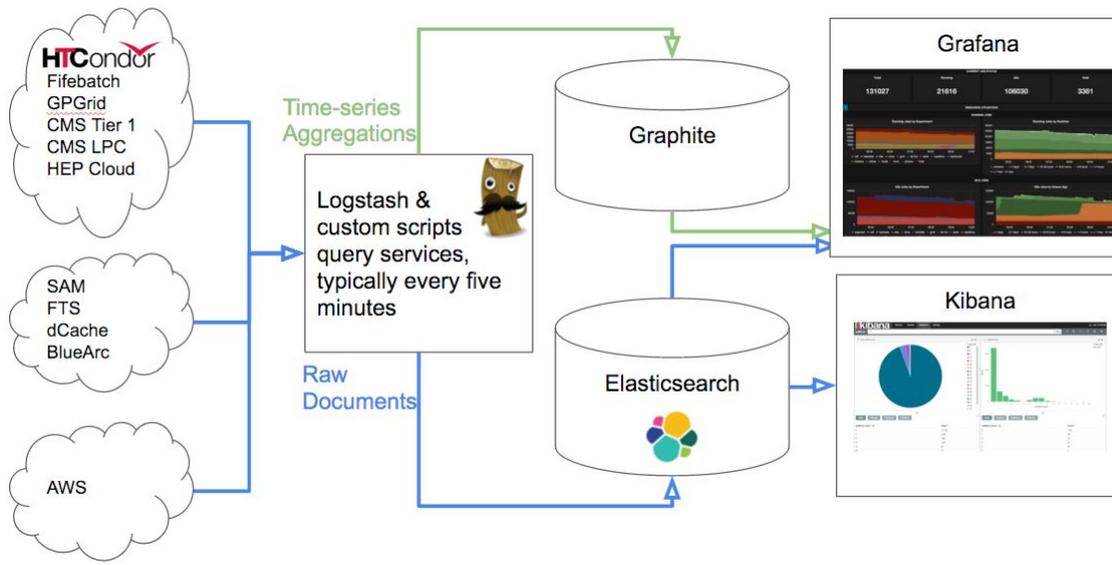
Elasticsearch, an Open Source product widely used by various companies as well as HEP groups (UChicago, CERN, CMS DAQ, etc), is used for the stats that have many to many relationships to each other, the number of categories, and relationships among categories, can't be determined in advance. We will use it to collect information from various logs, such as HTCondor event log, ifdh log, SAM and FTS logs. We are also planning to store all Gratia probe records in Elasticsearch.

### **9.0.3 Data Visualization Layer**

Grafana and Kibana has been chosen to serve as Data Visualisation Services for the Landscape Program. The main focus for Grafana is time series and graph panels. Grafana is easily extendable and provides a variety of panels with rich visualization options. There is built in support for many of the most popular time series data sources. It is more suitable for real time monitoring and provides a reasonable response time. Kibana will be used as a log analytics dashboard. Kibana enables us to analyse the collected data and see trends.

## **9.1. Architectural Diagram for User and Log Analytics Service for FIFE Experiments**

FIFEMon provides views of the collected data customized for a particular audience. This may be the data related to an individual user's activities, the data related to the activities of all the users on an individual experiment, or the data related to an individual service for example. It is very configurable drawing data from many sources to produce a holistic view of the status of the system.

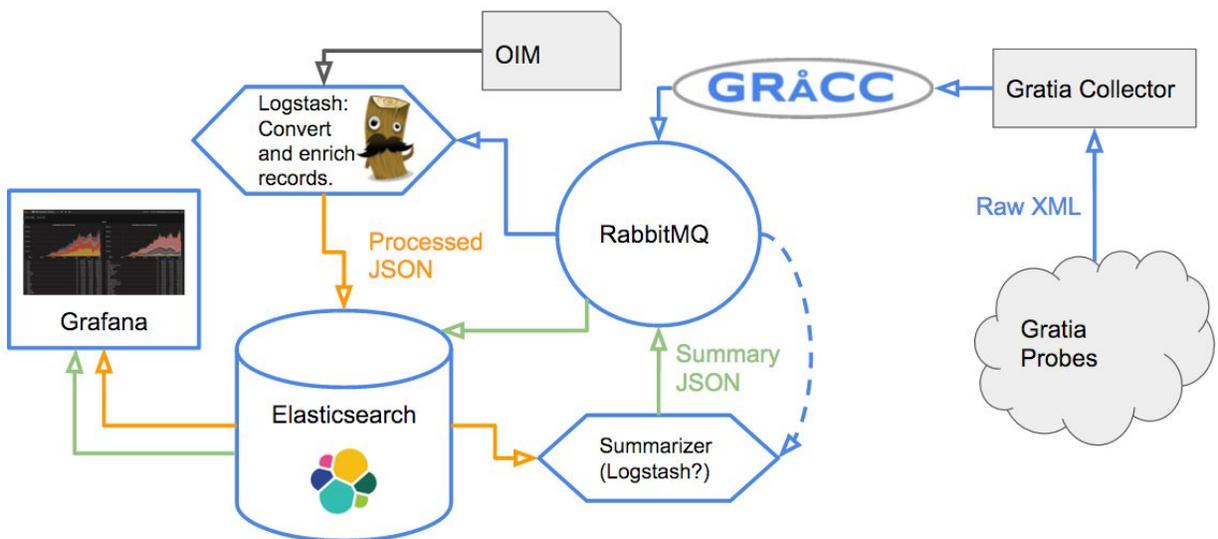


## Major Components:

- Custom Probes
  - Time-series data related to condor stats, AWS usage, status of data management services, etc.
    - Deployed on a host that has access to a services producing data, e.g HTCondor schedd, dCache InfoProvider, AWS.
  - Filtered logs from various services sending data to Elasticsearch cluster via Logstash clients.
    - Deployed on a host where service logs are available, eg. HTCondor head node with event logs, FTS and SAM station nodes.
- Open Source NOSQL repository
  - Graphite. Deployed on designated server with enough cores, memory and fast disk (preferably SSD).
  - Elasticsearch.
    - Deployed on several multicore nodes with 64GB RAM and fast disks. SSDs is preferable for indexing-heavy nodes.
- Open Source Visualization Dashboards
  - Grafana dashboards for time-series display.
    - Hardware requirements vary based on the quantity of nodes being monitored, amount of instances monitored and the frequency of monitoring. It is usually deployed on the same node as Graphite.
  - Kibana dashboard for analytic and creation of complicated queries and plot them as bar charts, line and scatter plots, pie charts, histograms.
    - No outstanding hardware requirements.

## 9.2. Architectural Diagram for GRACC Service

GRACC (GRid ACcounting Collector) Service is a collection of components for implementing resource usage accounting. It is a replacement of the Gratia accounting system. One of the goals is to replace a monolithic, “in-house” developed service with independent, mostly widely used open source components. The other goal is provide a flexible repository schema evolution for new data sources including memory consumption, network usage and many others. The service should be designed with long term maintenance in mind.



### Major Components:

- Collector
  - "Gratia-Compatible Collector" that acts as a transitional interface between the obsolete [Gratia](#) accounting collector and probes and the new GRACC accounting system.
  - It listens for bundles of records (as would be sent via replication from a Gratia collector or from a Gratia probe) on HTTP, processes the bundle into individual usage records, and sends those to RabbitMQ or another AMQP 0.9.1 broker.
- AMPQ Service
  - RabbitMQ, open source broker that supports HTTP, STOMP, AMQP 1.0 and others, enables software applications to connect and scale. Applications can connect to each other, as components of a larger application, or to user devices and data. Messaging is asynchronous, decoupling applications by separating sending and receiving data.
- Probes:

- Custom Gratia Probes that currently exists and will not be changed
- Additional Probes that could be developed for collecting Network, Condor Events and other information
- Replay daemons:
  - Daemons that listen and respond to requests for replays on a AMQP queue.
- Open Source Visualization Dashboards (GratiaWeb Replacement)
  - Grafana for time-series plots
  - Kibana for complicated queries
- Open Source NOSQL repository (MySQL Replacement)
  - Elasticsearch

### 9.3. Service and Site Availability Service

Check\_MK is an open source extension to the [Nagios](#) monitoring system that allows creating rule-based configuration using [Python](#). It comes with a set of system checks, a [mod\\_python](#) and [JavaScript](#) based web [user interface](#), and a module that allows fast access to the Nagios core. It is widely used in CD and will not be described here. We will continue development of check\_mk agents to monitor FIFEMon infrastructure and will configure them to report to existing check\_mk instance run by GCO.

### 9.4. Program Execution Architecture

Given the goals and the proposed technical architecture we propose a multi-project program execution architecture. Initially this consists of several projects and will be implemented in phases. Currently we have identified four projects that aim to address the requirements we have collected so far.

1. User and Log Analytics Service for FIFE (FIFEMON & FIFEMON-Analytics):
  - a. Allows users and support groups to monitor jobs, data transfers , and resource usage on HEP Cloud.
  - b. Collects various logs, allows to do monitoring, search, analysis, and visualization in real time.
2. Accounting (GRACC): Provides probes, repositories, and dashboard for historical information about user jobs, data transfers, storage usage.
3. Service and Site Availability Service: Collects status of the various services and Grid sites, allows to generate alarms and handle troubleshooting responses.

## 10. Appendix A: Justification for Open Source Software Selection

*“We were not pioneers ourselves, but we journeyed over old trails that were new to us, and with hearts open. Who shall distinguish?”*

*J. Monroe Thorington*

We are proposing to move from mostly in-house developed software to the 21st century open source products that have been proven to be flexible and scalable enough to be used widely by many organizations and communities. In HEP these tools are also becoming very popular and are running in production on multiple sites. Some examples are below.

### 10.1. ELK in HEP

1. ATLAS Analytics platform (running in CloudLab w/ indices backed up to MWT2).
  - a. Recent presentation could be found [here](#)
  - b. Installation:
    - 3B docs in 1084 indices spread over 9182 shards. Data size (single copy of it) ~3TB. Keep at least 2 copies. Growing ~2GB a day. They are indexing much more but some large indices from Rucio are deleted after 10 days. Updates at 3kHz (max 10kHz seen).
2. WLCG perfSONAR dashboard (global) and [regional federation dashboard \(MWT2\)](#)
3. [UChicago](#): Early The ES instance at UC uses 4 Dell R410 nodes with 3TB of storage on each and with most indices set to replicate data to all the data nodes. The master and client nodes are VMs. With this setup we're pretty overprovisioned in regards to cpu (load is usually under 4 with 12cores/24 w hyperthreading). The data nodes have 48GB of memory which is sufficient (ES should use no more than 32GB RAM per instance. The rest is used by the OS for caching.. ES has 12TB raw at Chicago (with 2x replication).
4. [CERN](#)
  - [Elasticsearch vs Oracle evaluation](#)
  - Used for Messaging service to monitor the status of their services. At the moment, there are more than two billion documents.
  - Will expand the usage of Elasticsearch to the three following areas:
    - Data transfer movements between the sites
    - Job processing
    - Status of the sites and services
- [dCache](#) (ELK)
- Tier-2 site for the ALICE collaboration at LHC (the Torino INFN CC) [is using ELK](#)
- [CMS DAQ](#)

- CMS has implemented a monitoring system for post-long shutdown 1 that complements the redesigned File-based Filter Farm and takes advantage of elasticsearch. It is able to provide a quasi-realtime full-detail insight into event processing information, using same mechanisms to inspect live run information as well as run history.
- OSG Connect [analytics](#)

## 11. Appendix B: Landscape and Other SCD Projects

There are numerous services that are developed or adapted by various teams in the Scientific Computing Division. Among these services are dCache, Enstore, SAM, FTS, ELog, IFBeam and many others. Some of these services are already reporting information relevant to FIFE off-line processing to FIFEMon and Gratia (GRACC). The access to this data via Landscape provides many benefits for stakeholders, namely access to just one service with easy navigation between dashboards, possibility to correlate data, and similar look and feel. There are other services that are still under development, such as POMS (Production Operation Management Service) and CI (Continuing Integration Service), that could benefit significantly by being incorporated into Landscape framework.