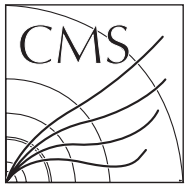
Available on CMS information server

CMS CR -2018/390



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

27 November 2018

Presentation layer of CMS Online Monitoring System

Mantas Stankevicius for the CMS Collaboration

Abstract

The Compact Muon Solenoid (CMS) is one of the experiments at the CERN Large Hadron Collider (LHC). The CMS Online Monitoring system (OMS) is an upgrade and successor to the CMS Web-Based Monitoring (WBM) system, which is an essential tool for shift crew members, detector subsystem experts, operations coordinators, and those performing physics analyses.. CMS OMS is divided into aggregation and presentation layers. Communication between layers uses RESTful JSON API compliant requests. The aggregation layer is responsible for collecting data from heterogeneous sources, storage of transformed and pre-calculated (aggregated) values and exposure of data via the RESTful API.The presentation layer displays detector information via a modern, user-friendly and customizable web interface. The CMS OMS user interface is composed of a set of cutting-edge software frameworks and tools to display non-event data to any authenticated CMS user worldwide. The web interface tree-like component structure comprises (top-down) workspaces, folders, pages, controllers and portlets. A clear hierarchy gives the required flexibility and control for content organization. Each bottom element instantiates a portlet and is a reusable component that displays a single aspect of data, like a table, a plot, an article, etc. Pages consist of multiple different portlets and can be customized at run-time by using a drag-and-drop technique. This is how a single page can easily include information from multiple online sources. Different pages give access to a summary of the current status of the experiment, as well as convenient access to historical data.This paper describes the CMS OMS architecture, core concepts and technologies of the presentation layer.

Presented at *CHEP 2018 Computing in High-Energy Physics 2018*

Presentation layer of CMS Online Monitoring System

Jean-Marc André[5], *Ulf* Behrens[1], *James* Branson[4], *Philipp* Brummer[2,12], *Sergio* Cittolin[4], *Diego* Da Silva Gomes[2], *Georgiana-Lavinia* Darlea[6], *Christian* Deldicque[2], *Zeynep* Demiragli[6], *Marc* Dobson[2], *Nicolas* Doualot[5], *Samim* Erhan[3], *Jonathan Richard* Fulcher[2], *Dominique* Gigi[2], *Maciej* Gładki[2], *Frank* Glege[2], *Guillelmo* Gomez-Ceballos[6], *Jeroen* Hegeman[2], *André* Holzner[4], *Mindaugas* Janulis[9,11], *Michael* Lettrich[2], *Audrius* Mečionis[5,10], *Frans* Meijers[2], *Emilio* Meschi[2], *Remigius K* Mommsen[5], *Srecko* Morovic[5], *Vivian* O'Dell[5], *Luciano* Orsini[2], *Ioannis* Papakrivopoulos[7], *Christoph* Paus[6], *Petia* Petrova[2], *Andrea* Petrucci[8], *Marco* Pieri[4], *Dinyar* Rabady[2], *Attila* Rácz[2], *Valdas* Rapševičius[5,13], *Thomas* Reis[2], *Hannes* Sakulin[2], *Christoph* Schwick[2], *Dainius* Šimelevičius[2,10], *Mantas* Stankevičius[5,10*], *Cristina* Vazquez Velez[2], *Christian* Wernet[2], and *Petr* Zejdl[5,11]

[1]DESY, Hamburg, Germany
[2]CERN, Geneva, Switzerland
[3]University of California, Los Angeles, Los Angeles, California, USA
[4]University of California, San Diego, San Diego, California, USA
[5]FNAL, Batavia, Illinois, USA
[6]Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
[7]Technical University of Athens, Athens, Greece
[8]Rice University, Houston, Texas, USA
[9]Vilnius University, Vilnius, Lithuania
[10]Also at Vilnius University, Vilnius, Lithuania
[11]Also at CERN, Geneva, Switzerland
[12]Also at Karlsruhe Institute of Technology, Karlsruhe, Germany
[13]Also at Vilnius University, Institute of Computer Science, Vilnius, Lithuania

Abstract. The Compact Muon Solenoid (CMS) is one of the experiments at the CERN Large Hadron Collider (LHC). The CMS Online Monitoring system (OMS) is an upgrade and successor to the CMS Web-Based Monitoring (WBM) system, which is an essential tool for shift crew members, detector subsystem experts, operations coordinators, and those performing physics analyses. CMS OMS is divided into aggregation and presentation layers. Communication between layers uses RESTful JSON:API compliant requests. The aggregation layer is responsible for collecting data from heterogeneous sources, storage of transformed and pre-calculated (aggregated) values and exposure of data via the RESTful API.

The presentation layer displays detector information via a modern, user-friendly and customizable web interface. The CMS OMS user interface is composed of a set of cutting-edge software frameworks and tools to display non-event data to any authenticated CMS user worldwide. The web interface tree-like component structure comprises (top-down): workspaces, folders, pages, controllers and portlets. A clear hierarchy gives the required flexibility and control for content organization. Each bottom element instantiates a portlet and is a reusable component that displays a single aspect of data, like a table, a plot, an article,

*Corresponding author: Mantas.Stankevicius@cern.ch

etc. Pages consist of multiple different portlets and can be customized at run-time by using a drag-and-drop technique. This is how a single page can easily include information from multiple online sources. Different pages give access to a summary of the current status of the experiment, as well as convenient access to historical data.
This paper describes the CMS OMS architecture, core concepts and technologies of the presentation layer.

1 Introduction

Since the inception of CMS the Web-Based Monitoring (WBM) activity was the one that provided tools for run-time and retrospective monitoring of the detector. During the first decade of data taking the WBM has accumulated experience and tools that provided efficient detector monitoring [1]. In order to ensure longterm support of the manpower and technical resources for the monitoring tools required for CMS, in 2015 it was decided to re-designed the core functionality of former WBM into the CMS Online Monitoring System (OMS) into two layers: one for the aggregation layer and another for presentation. Framework and initial content development took place throughout the 2017 and the first production version announced in February of 2018. It is planned that OMS will have a full functionality starting Run III (2021, fully replacing WBM) while active development must converge during 2019.

2 Architecture

Following best practices CMS OMS is designed to have two separate layers: aggregation layer and presentation layer. This architecture makes a clear separation of GUI and data in the system and makes it possible to use/query data via other automated means, like CLI, scripts, other GUI services. Not having this separation was considered a major draw-back in the former WBM. Layers communicate via RESTful API.

The aggregation layer [2] features multiple specialized endpoints (microservices), each implementing the specialized SQL query. On request, the appropriate application executes associated optimized queries on Oracle database, fetches data, serializes data along with metadata into JSON format and returns to the client. Aggregation layer is implemented as a standalone Java application running on Katharsis framework [3]. The application interface follows the specification of JSON:API [4].

3 Presentation Layer

The presentation layer itself consists of two separate applications: metadata API and web application.

3.1 Metadata API

Metadata API provides the back-end application service and access to persistent storage for the portal configuration. Service provides a JSON:API [4] compliant interface to access, store and update interface objects. It is a python application running on Flask [5] with database access via SQLAlchemy framework [6]. Data is stored on Oracle relational database. Application code follows pep8 coding guidelines [7]. It is a stateless and scalable application. Content synchronization and integrity is ensured at the database level.

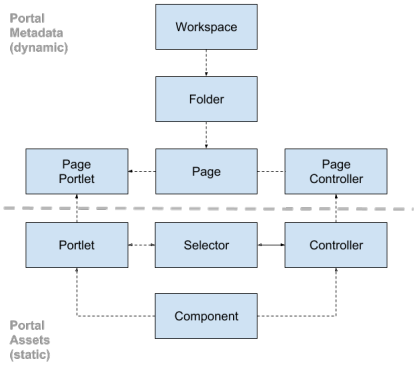


Figure 1. Major persistent objects of presentation layer

3.2 Web Application

The Web Application uses state-of-the-art technologies like ReactJS [8], Redux [9], MaterialUI [10], Highcharts [11]. The application is lightweight, responsive, interactive and user friendly.

4 Content organization

Website content is organized in a hierarchical manner by enclosing the structure into the set of persistent objects (see Fig. 1). Objects are divided into two groups: portal metadata (dynamic) objects that can be managed by GUI administrators, and portal assets (static) objects that are added to the portal with software updates, i.e. code base.

Workspace is a top-level container which contains folders and defines organizational unit, like experiment project, sub-detector, etc. For example: CMS, DAQ, Pixel, CSC. Each workspace can contain multiple folders which are the mid-level containers for grouping pages. Page is a lowest level content container which provides the actual display of content to the end-users. A page fills a browser window and is composed of a single optional controller and one or more portlets. For example, Run Summary, Fill Report, RBX (Read-Out Box) Plots, etc. All above objects, workspace to page, are the dynamic portal metadata which are stored within the metadata API persistent storage and can be created/edited/deleted via portal management GUI section.

Controller is a filtering or selection widget which allows the user to apply selection on the whole page, specifically on portlets within the page. Controller provides selected data object (selection) to each page portlet and refreshes it. For example, Run selection, Run Range selection, Fill number, etc. Portlet is a single content presentation object which presents a single well-defined aspect of information, i.e. a single plot, a table, a set of links, etc., for example, Main Run information table, Fill Analysis plot, RBX (Read-Out Box) FNAL (Fermi Laboratory) analysis IV (Current-Voltage) curve. Each portlet can act as controller too where it provides a selection to other portlets. In this way it is possible to establish the master – detail relationship from one portlet to the other within a single page. The component is the last object in the set which represents a reusable GUI element, like simple button, input box or a more sophisticated - data table, dynamic plot, etc. Controllers, portlets and components are

represented as Javascript/CSS files within the application code base. All the above mentioned portal objects are described in more details below.

4.1 Workspace

Workspace is the top level content container which is predefined and comes from the metadata repository. At the time of writing, OMS had one central workspace - CMS. At the level of workspaces the administration role is being checked: individual CERN users or/and e-groups can be added into the set of workspace administrators. They are able to manage the lower level dynamic objects of the presentation layer.

4.2 Folder

Folders are the intermediate grouping objects that contain one or more pages. At the time of writing OMS had 4 folders: Index, Fills, Runs and Triggers. This sequence follows the actual data taking workflow of LHC and an experiment: LHC starts a *Fill* by inserting bunches into the accelerator then detector starts a *Run* to record event data each initiated by the *Trigger*. Folders can be added, edited, arranged or removed by workspace administrators. Folders are identified by the unique (within workspace) name or title which directly translates into the URL path and optional description.

4.3 Page

Page is the final dynamic container which contains an optional controller and one or more portlets (see the list in Table 1). As well as folders, pages can be added, edited, arranged or removed by workspace administrators. Pages are identified by the unique within workspace name or title which directly translates into the URL path and optional description. Optional controller can be chosen from the list of available controllers while portlets can be added afterwards. Page controller is not visible by default but can slide-in from the portal header by clicking on the page title or selection abbreviation. Once the page is created, the workspace administrator can add/edit/remove portlets by using the drag-and-drop technique from within the main portal display. Once portlets were added to the page and all the content was rendered, the administrator can re-arrange the page layout by dragging/re-sizing portlets. This real-time layout editing technique allows administrators to adjust the portlet sizes to the actual content.

4.4 Controller

Controller is a static portal object which can be edited only in the code base and the update requires to re-deploy the application. The list of main controllers is presented in Table 2. Each page can have no more than one controller which, on the other hand, can be used in many pages. The controller is the special selection widget which maintains the clear defined interface to provide the selection values to page portlets. The controller has two states - open, all the selection components, like input boxes, drop-down boxes, buttons, etc. are visible on the page (see Figure 3), and closed, all the selection components are hidden and only the selection snippet is visible. This open-closed technique allows to provide sophisticated widgets for value selection as well as hide it once the selection is done and the focus is on the data portlets instead. Controllers are based on the re-usable controller components which are discussed below. Eventually, a controller allows the user to apply a selection on the page, i.e. Run range, date range, Fill number, boolean operators. All selections in a controller are translated into a shareable URL with parameters.

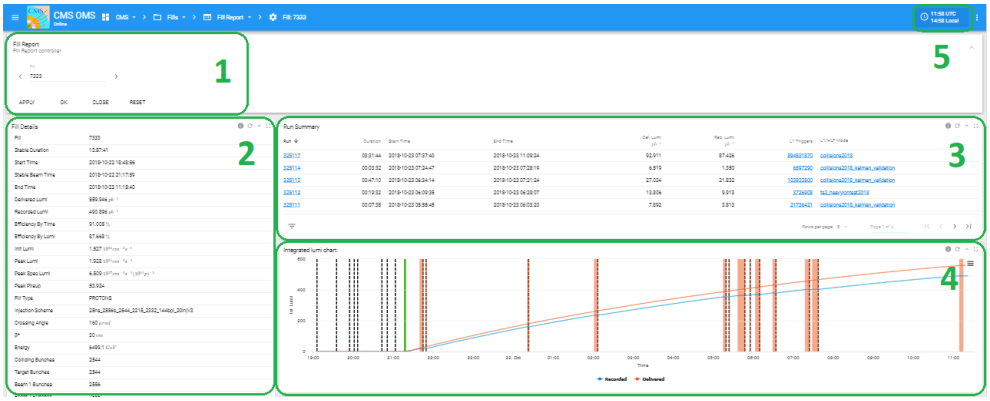


Figure 2. Fill Report page. 1 - open controller, 2 - vertical Data table, 3 - horizontal Data table, 4 - dynamic Chart, 5 - UTC and local time

Table 1. List of OMS pages (as of paper publication time)

Folder	*Title*	*Description*
Index	Index	Start page. Features introductory and overview content.
Fills	Fill Summary	Brief information about LHC Fills.
Fills	Fill Report	Detailed information about the single LHC Fill (various tables, plots, etc.) and table with links to Runs.
Runs	Run Summary	Brief information about CMS Runs.
Runs	Run Report	Detailed information about the single CMS data taking Run (tables, plots, etc.), links to other pages.
Runs	Lumisections	Detailed information about each lumisection.
Runs	Prescaling	List of prescale sets of each Algorithm Trigger.
Triggers	L1 Current Rates	Displays Trigger configuration, rate and count, Overall and Beam Active Deadtimes.
Triggers	L1 Trigger Rates	Same as L1 Current Rates, but for finished CMS Run.
Triggers	L1 Algorithm Report	Detailed information about single L1 Algorithm Trigger, rate and count per lumisection in tabular form and chart.
Triggers	Trigger Report	Full information about Trigger: menus, keys, etc.

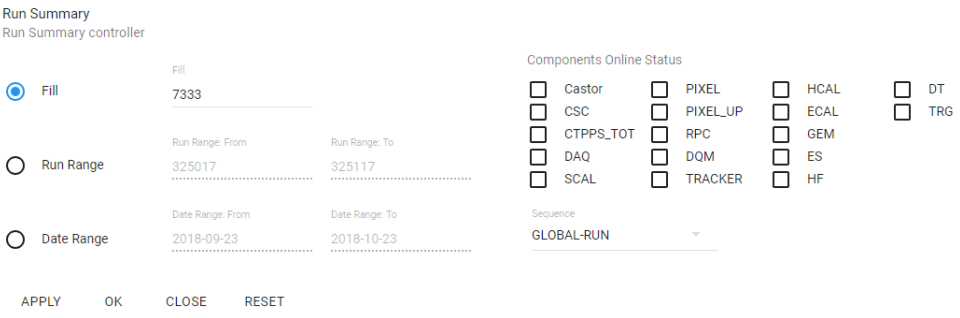


Figure 3. Run Summary controller. Selectors: LHC Fill, CMS Run range, date range, sequence, included subsystems

Table 2. List of main OMS controllers (13 in total as of paper publication time)

Title	*Description*
Fill Summary	Filtering by LHC Fill range, date range, era, runtime type.
Run Summary	Filtering by LHC Fill, CMS Run range, date range, sequence, subsystems.
Run Report	Filtering by CMS Run number.
Fill Report	Filtering by LHC Fill number.
L1 Algo Trigger	Filtering by Bit and CMS Run number.

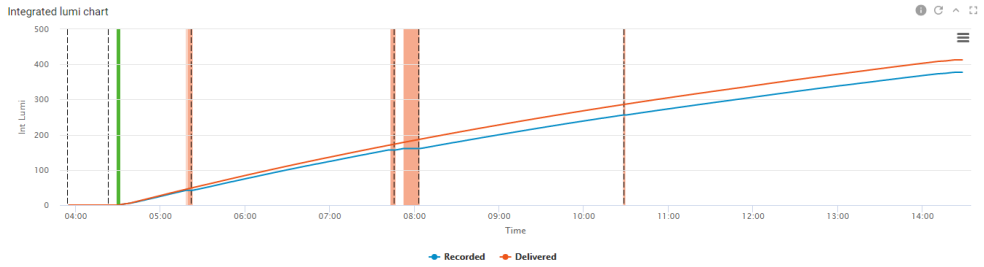


Figure 4. Integrated Luminosity Chart. First bar - start of LHC stable beam, sequent bars - downtimes, black dashed lines - start of new CMS runs

4.5 Portlet

Portlet, as well as controller, is a static portal object which can be edited only in the code base and the update requires to re-deploy the application. Each page can have more than one portlet which can be used in many pages as well. The portlet is the ultimate content container which displays a single well-defined aspect of the information, like a specific table or plot. Portlet objects maintain a specific interface which allows a page to propagate selection (controller values), refresh, minimize, maximize, edit and so on. This makes it possible to re-use portlets in many different pages as well as provide the unified and recognizable functionality for the different content portlets. Portlets are based on the re-usable portlet components which are discussed below. Every portlet has basic functionality: description tool-tip, collapse(minimize), full-screen and refresh. There are many ways to refresh a portlet: refresh the whole page, apply new selection in controller, click a button or set a timer. Portlets can fetch data from multiple sources, combine it and visualize. Data from four endpoints is used in order to construct Figure 4.

By definition a portlet is an isolated component, however it is possible to assign a group number to a portlet. Multiple portlets with the same group number can communicate with each other within the group scope, that works like "master-details" pattern (see Figure 5). The table on the left displays several quantities of last lumisection (sub-section of a run during which time the instantaneous luminosity is unchanging). A user can select multiple checkboxes on the table and these quantities are plotted at the right chart over time or lumisections (optional).

4.6 Selector

A selector is a simple portal object which is used for synchronization between controller and portlets as well as master and detail portlets. It is identified by the name which must be unique within the portal. If selector is attached to a certain controller than we can state that controller

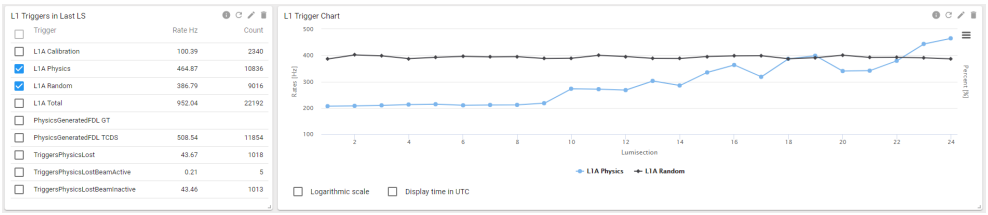


Figure 5. Master-details

```
{
  "label": "Run",
  "name": "run_number",
  "props": {
    "href": "/cms/runs/report?cms_run=%",
    "params": ["run_number"],
    "type": "link"
  }
}
```

```
{
  "label": "Duration",
  "name": "duration",
  "props": {
    "type": "duration"
  }
}
```

```
{
  "label": "Del. Lumi",
  "name": "delivered_lumi",
  "show_units": true,
  "props": {
    "ndigits": 3,
    "type": "float"
  }
}
```

Figure 6. Portlet/component configuration

"speaks this language". While if the same selector is attached to the portlet than it can be said that portlet "understands this language". Thus a selector acts as common denominator for the controller and multiple portlets to co-exist in the same page.

4.7 Components

Main page objects - controllers and pages are driven by reusable components and custom configuration (see list in Table 3). Components are included into the code base and implemented by using Javascript/CSS. Components can define specific configuration requirements i.e. data URL endpoint, page size, image URL, image size, etc. Some endpoints allow the content to be added as in the frame of configuration, i.e. portlet which displays links (URLs) to other resources reads a list of links from the configuration. Example configuration is in Figure 6. First block dynamically generates a link using pattern, second block converts duration in seconds into human readable format (hh:mm:ss), third block formats received value and sets no more than 3 digits of precision.

5 Conclusions

This paper described the architecture and implementation of the Online Monitoring System (OMS). It provides a clear separation between data retrieval (querying and formatting, aggregation API) and presentation (formatting and display, presentation) layers. This separation allows the two layers to be developed and deployed independently as long as the specified interface is maintained.

Hierarchical content organization and the metadata API of the Presentation Layer provides flexibility in content display organization by rapid display re-factoring without code changes. In addition, the presentation layer object schema makes it possible to reuse controllers, portlets and components on any level of organization thus providing a consistent look-and-feel and functionality throughout the application by reducing the need for development. Administration access control on workspace level is well leveraged for the CMS

Table 3. List of main generic OMS components (around 20 in total as of paper publication time)

Title	*Description*
Datatable	Data table is most advanced and most used component in OMS. Data table can be configured to display vertically (key:value) or horizontally (multiple rows vs columns). Horizontal data table provides extensive functionality to manipulate dataset: sort by column, add/remove columns, filter on column, paginate and change page size. Numerical values within a page can be aggregated by one of the functions: avg/sum/min/max. Units are displayed in a header row of a data table in KaTeX [12] format. Under certain conditions column might have different units within a page, in this case data table displays both value and units in the same cell.
Image	Displays one or more images. Only one image is displayed at a time, carousel allows iterate over multiple images.
Links	Displays both static and dynamic links.

experiment needs as most of the performance and control management is done on the group or project level.

References

[1] J.A. Lopez-Perez, K. Maeshima, W. Badgett, U. Behrens, I. Chakaberia, Y. Jo, S. Maruyama, J. Patrick, V. Rapsevicius, A. Soha et al., J. Phys. : Conf. Ser. **898**, 092040. 8 p (2017)

[2] C. Wernet, C. Zirpins, A. Petrucci, *Unifying access to data from heterogeneous sources through a RESTful API using an efficient and dynamic SQL-query builder* (2017), `http://cds.cern.ch/record/2644673`

[3] *Katharsis-framework*, `https://github.com/katharsis-project/katharsis-framework`

[4] *jsonapi.org*, `https://jsonapi.org/`

[5] *Flask web development*, `http://flask.pocoo.org/`

[6] *The python sql toolkit and object relational mapper*, `https://www.sqlalchemy.org/`

[7] *Pep 8 - style guide for python code*, `https://www.python.org/dev/peps/pep-0008/`

[8] *React - a javascript library for building user interfaces*, `https://reactjs.org/`

[9] *Redux - predictable state container for javascript apps*, `https://redux.js.org/`

[10] *Material-ui - react components that implement google's material design*, `https://material-ui.com/`

[11] *Highcharts - make your data come alive*, `https://www.highcharts.com/`

[12] *Katex. the fastest math typesetting library for the web*, `https://katex.org/`