Available on CMS information server **CMS CR -2017/039**

The Compact Muon Solenoid Experiment

Conference Report

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

16 February 2017 (v2, 17 February 2017)

New operator assistance features in the CMS Run Control System

Jean-Marc Andre[5], Ulf Behrens[1], James Branson[4], Philipp Brummer[2], Olivier Chaze[2], Sergio Cittolin[4], Cristian Contescu[5], B.G. Craigs[2], Georgiana-Lavinia Darlea[6], Christian Deldicque[2], Zeynep Demiragli[6], Marc Dobson[2], Nicolas Doualot[5], Samim Erhan[3], Jonathan F. Fulcheri[2], Dominique Gigi[2], Michail Gładki[2], Frank Glege[2], Guillelmo Gomez-Ceballos[6], Jeroen Hegeman[2], Andre Holzner[4], Mindaugas Janulis[2], Raúl Jimenez-Estupiñá[2], Lorenzo Masetti[2], Frans Meijers[2], Emilio Meschi[2], Remigius K. Mommsen[5], Srecko Morovic[2], Vivian O'Dell[5], Luciano Orsini[2], Christoph Paus[6], Petia Petrova[2], Marco Pieri[4], Attila Racz[2], Thomas Reis[2], Hannes Sakulin[2], Christoph Schwick[2], Dainius Simelevicius[2], Petr Zejdl[5]

Abstract

The Run Control System of the Compact Muon Solenoid (CMS) experiment at CERN is a distributed Java web application running on Apache Tomcat servers. During Run-1 of the LHC, many operational procedures have been automated. When detector high voltages are ramped up or down or upon certain beam mode changes of the LHC, the DAQ system is automatically partially reconfigured with new parameters. Certain types of errors such as errors caused by single-event upsets may trigger an automatic recovery procedure. Furthermore, the top-level control node continuously performs cross-checks to detect sub-system actions becoming necessary because of changes in configuration keys, changes in the set of included front-end drivers or because of potential clock instabilities. The operator is guided to perform the necessary actions through graphical indicators displayed next to the relevant command buttons in the user interface. Through these indicators, consistent configuration of CMS is ensured. However, manually following the indicators can still be inefficient at times. A new assistant to the operator has therefore been developed that can automatically perform all the necessary actions in a streamlined order. If additional problems arise, the new assistant tries to automatically recover from these. With the new assistant, a run can be started from any state of the subsystems with a single click. An ongoing run may be recovered with a single click, once the appropriate recovery action has been selected. We review the automation features of the CMS run control system and discuss the new assistant in detail including first operational experience.

[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, Chicago, Illinois, USA

[6] Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

New operator assistance features in the CMS Run Control System

J-M Andre[4], U Behrens[2], J Branson[3], P Brummer[1,a], O Chaze[1], S Cittolin[3], C Contescu[4], B G Craigs[1], G-L Darlea[5], C Deldicque[1], Z Demiragli[5], M Dobson[1], N Doualot[4], S Erhan[6], J R Fulcher[1], D Gigi[1], M Gładki[1], F Glege[1], G Gomez-Ceballos[5], J Hegeman[1], A Holzner[3], M Janulis[1,b], R Jimenez-Estupiñán[1], L Masetti[1], F Meijers[1], E Meschi[1], R K Mommsen[4], S Morovic[1], V O'Dell[4], L Orsini[1], C Paus[5], P Petrova[1], M Pieri[3], A Racz[1], T Reis[1], H Sakulin[1,7], C Schwick[1], D Simelevicius[1,b], M Vougioukas[1], P Zejdl[4,c]

[1] CERN, Geneva, Switzerland
[2] DESY, Hamburg, Germany
[3] University of California, San Diego, San Diego, California, USA
[4] FNAL, Chicago, Illinois, USA
[5] Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
[6] University of California, Los Angeles, Los Angeles, California, USA
[a] Now at Karlsruhe University of Applied Sciences, Karlsruhe, Germany
[b] Also at Vilnius University, Vilnius, Lithuania
[c] Also at CERN, Geneva, Switzerland

E-mail: Hannes.Sakulin@cern.ch

Abstract. During Run-1 of the LHC, many operational procedures have been automated in the run control system of the Compact Muon Solenoid (CMS) experiment. When detector high voltages are ramped up or down or upon certain beam mode changes of the LHC, the DAQ system is automatically partially reconfigured with new parameters. Certain types of errors such as errors caused by single-event upsets may trigger an automatic recovery procedure. Furthermore, the top-level control node continuously performs cross-checks to detect sub-system actions becoming necessary because of changes in configuration keys, changes in the set of included front-end drivers or because of potential clock instabilities. The operator is guided to perform the necessary actions through graphical indicators displayed next to the relevant command buttons in the user interface. Through these indicators, consistent configuration of CMS is ensured. However, manually following the indicators can still be inefficient at times. A new assistant to the operator has therefore been developed that can automatically perform all the necessary actions in a streamlined order. If additional problems arise, the new assistant tries to automatically recover from these. With the new assistant, a run can be started from any state of the subsystems with a single click. An ongoing run may be recovered with a single click, once the appropriate recovery action has been selected. We review the automation features of CMS Run Control and discuss the new assistant in detail including first operational experience.

[7] To whom any correspondence should be addressed.

1. Introduction

The Run Control System [1,2,3,4] of the Compact Muon Solenoid (CMS) [5,6] experiment at CERN's Large Hadron Collider (LHC) is a distributed Web Application running on a set of Apache Tomcat servers. It allows users to define a hierarchical structure of control nodes, called Function Managers (FMs). Function managers are developed in Java and are dynamically loaded into the web application upon the start of a configuration [7]. Function Managers are controlled through a web-browser. During global data taking, all data-taking operations are controlled through the top-level control node, also called the Level-0 FM as illustrated in figure 1. Hardware access and transport of data are handled by the C++ based XDAQ online software [8]. Monitoring services collect monitor data from the XDAQ and run control layers and make them available to monitor clients, error & alarm panels and an expert system. A separate control system, the Detector Control System (DCS) [9] controls detector voltages, gas flows etc. DCS is in communication with the LHC control system.

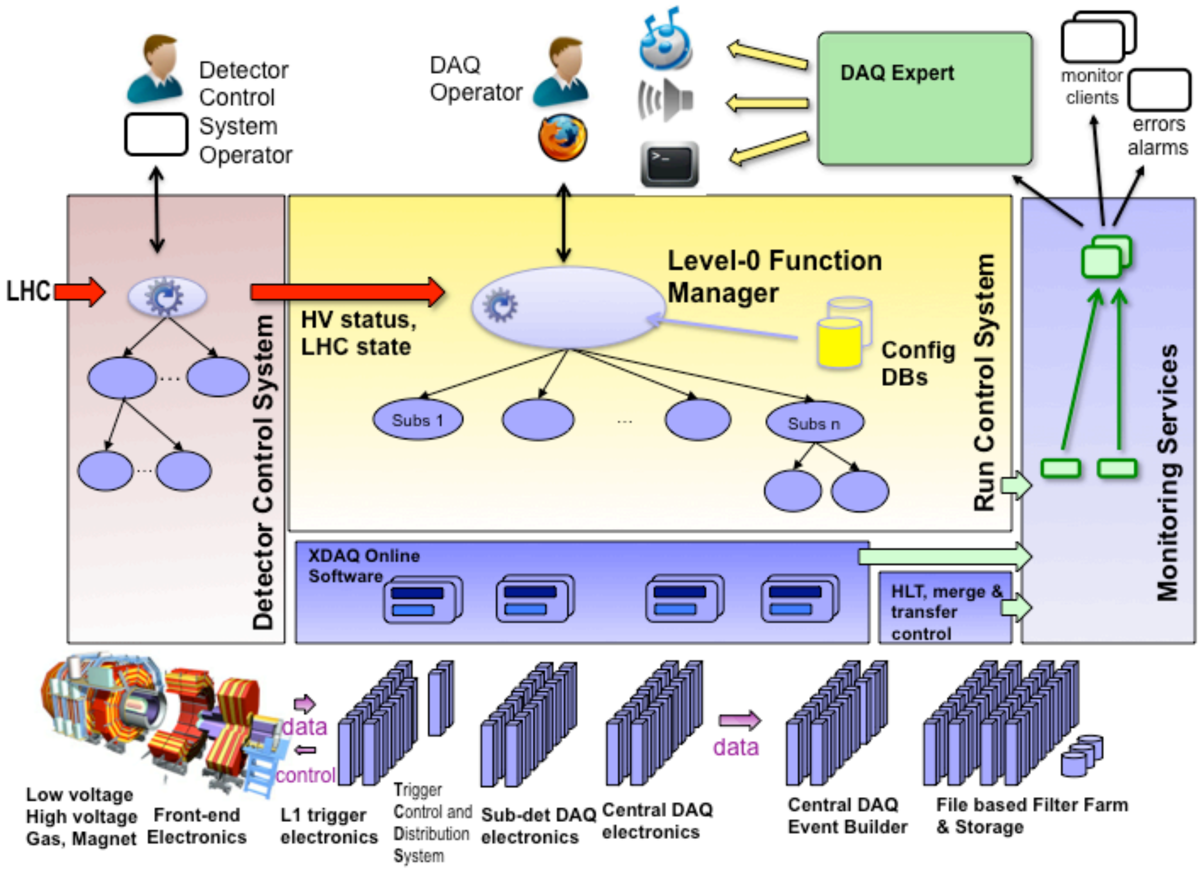


Figure 1. Overview of the CMS Online Systems.
Subs: Subsystem. HLT: High Level Trigger

2. Review of Operator Assistance Features

At the start of data taking with CMS in 2008 the top-level control node provided full control of the global state machine, control of individual subsystems and control of configuration keys applicable to various subsystems. All required operations were available to the operator, but the operator had to know how to choose compatible configuration keys, when to perform actions on individual subsystems and how these actions depended on each other. Operation of the experiment required a lot of detailed knowledge of the system and was rather error prone. Since then, many operator assistance features have been added to the top-level control node (Level-0 Function Manager) of the run-control system in order to ease the job of the operator and avoid any possibilities for errors. We briefly review these features, below. For a more in-depth description we refer the reader to [10].

2.1. Improved Configuration Handling
In the area of configuration handling, configuration keys were combined by introducing first a combined trigger key for the first-level and high-level trigger and later introducing a combined run-mode which ties together compatible sets of all configuration keys for all sub-systems needed for certain modes of data taking.

2.2. Guidance System
A guidance system was introduced in order to ensure consistent configuration of the experiment. It continuously checks whether all components are configured with the selected configuration, which is typically implied by the over-all run mode described above. The guidance system resolves configuration keys in a number of databases to detect if underlying definitions of keys have been updated. The system also ensures that configuration changes are applied to subsystems in the correct order. This is particularly important if the clock source of the experiment is changed or the experiment needs to recover from LHC clock instabilities. The guidance system indicates necessary actions to the operator by flashing indicator lights next to the corresponding sub-system action buttons in the GUI.

2.3. Automatic actions in response to detector and LHC state changes
When detector high voltages are ramped during preparation for data taking or just when stable beams have been declared, certain actions have to be performed through run-control. In the silicon pixel subsystem, gains need to be adjusted while in the silicon strips subsystem payload suppression needs to be turned off (payload-suppression is needed due to noise when high voltages are off). In order to ensure that these actions always take place, the top-level control node receives information about the LHC state and about detector high voltages from the CMS detector control system. If a run is ongoing while the LHC or high voltage change state, an automatic action is be triggered that pauses the run, propagates updated settings to subsystems and resumes the run. The information about the LHC state is also used to decide when a clock recovery needs to be performed (The LHC clock is guaranteed to be stable only in certain beam modes.) and to automatically select the appropriate run mode (for example: collisions, cosmics, circulating beam) for a certain data taking condition.

2.4. Automatic Error Recovery for 'Soft' Errors
As LHC luminosity increased towards the end on Run-1 (2011 onwards), frequent 'soft' errors occurred in various sub-systems which in some cases could be traced to single-event upsets in the readout electronics caused by increased irradiation. Recovery initially required stopping the run, reconfiguring the affected sub-system and restarting a run, a very time consuming operation. A new automatic recovery mechanism was therefore designed in which sub-systems locally detect a 'soft' error condition and signal it to the to the top-level control node by changing their state. The top-level control node then executes a recovery sequence that consists of: pausing the run, issuing a newly defined recovery transition to the requesting and subsystem, resynchronizing the system and resuming. Additional subsystems may register to receive the recovery transition in order to perform preventive actions in the shadow of a recovery.

3. The new Level-0 Automator
While the Soft Error Recovery covers frequent and well-understood errors, there remains a category of errors that are not or cannot be recovered by Soft Error Recovery because they are rare, not well understood or they cannot be diagnosed locally. This latter category of errors currently has to be dealt with by the shift crew according to instructions that describe how to diagnose the error and that prescribe a recovery procedure. Typically this procedure consists of 1) stopping the run, 2) re-configuring or re-initializing a subsystem, 3) starting a new run. Our long-term goal is to also automate recovery for this second category of errors. An automated recovery can be factored into two largely independent steps: diagnosing the error situation and executing the recovery procedure. The

diagnosis can be performed by an expert system based on the observables of the one-line data-flow monitoring. During Run-1, we successfully used the Perl-based DAQ Doctor [10] that gave advice to the shift crew. For Run-2 we are developing a new tool written in Java, called DAQ Expert. At the time of writing the new tool successfully recognizes most error situations and prescribes the appropriate recovery action.
In the present paper we want to focus on the second step of the recovery – the execution of the recovery procedure. This second step consists of executing a sequence of commands to the global state machine or to individual sub-systems. In addition to just following the prescribed steps, the recovery should also re-configure / re-initialized subsystems if indicated by the guidance system in order to ensure consistent configuration. Moreover, if subsystems fail during any step of the recovery action, a number of attempts should be made to recover them automatically – if possible in the shadow of other necessary steps. Just like a very experienced operator, the software orchestrating the recovery should have knowledge of the interdependency of transition actions of all subsystems in order to be able to execute actions in the optimal order and to parallelize actions wherever possible. Since the execution of the recovery procedure requires fast feedback on the state of sub-systems, we decided to implement it within the run control system rather than in the external expert tool that receives monitoring data with a certain latency.

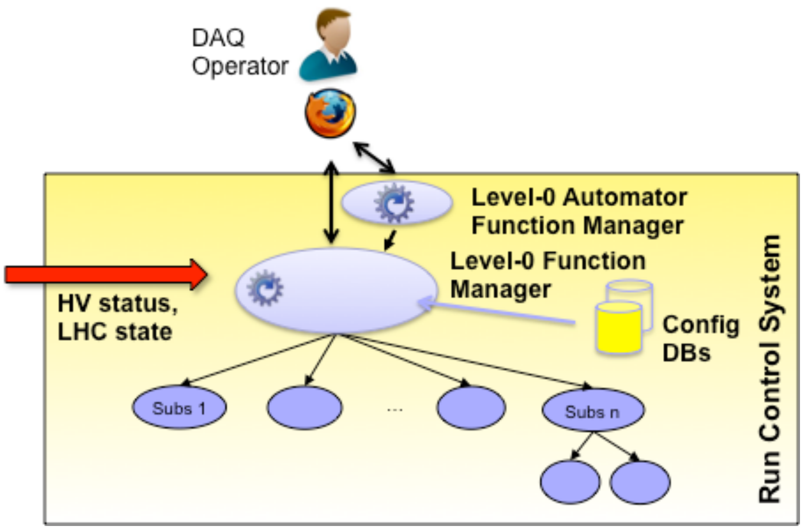
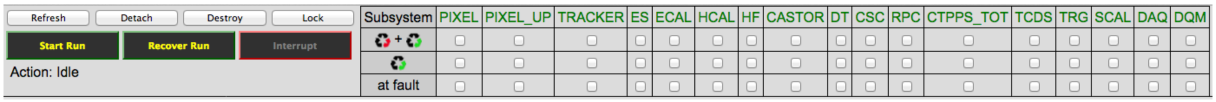


Figure 2. Level-0 Automator Function Manager in the FM hierarchy.

As the recovery procedure acts on the system in the way an experienced operator would, we chose to implement it in a new Function Manager called the *Level-0 Automator FM* that is located on top of the hierarchy – between the operator and the Level-0 FM – as shown in figure 2. This also has the advantage to keep the complexity in a separate entity, which is optional and can be deployed with minimum risk to the production system. The new function manager fulfils all the requirements discussed above. Its user-interface – shown in figure 3 – allows defining a recovery procedure by selecting which subsystems need re-configuration or re-initialization. Clicking the 'Recover' button then triggers the recovery procedure. Alternatively, the recovery can also be triggered by sending the recovery instructions to a SOAP interface. This interface will in future be used by the DAQ Expert tool. Since the Level-0 Automator knows how to get the system to running state from any state of the sub-systems it can also be used by the operators to start a run in the most streamlined manner. To this end a 'Start' button is available that will not stop any on-going run but otherwise triggers the same actions as the 'Recover' button. Since the Level-0 Automator performs actions that so far have been done by the operator, it is important to record the history of these actions and make it accessible to the operator. We therefore developed a timeline viewer, which shows the evolution over time of the global and all subsystem states and also shows all actions whether triggered by the Level-0 Automator or triggered manually by the operator. Figure 4 shows the timeline for the recovery while figure 5 shows the timeline for the start of a run.



Action: Idle

Subsystem	PIXEL	PIXEL_UP	TRACKER	ES	ECAL	HCAL	HF	CASTOR	DT	CSC	RPC	CTPPS_TOT	TCDS	TRG	SCAL	DAQ	DQM
♻ + ♻	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
♻	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
at fault	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Figure 3. GUI of the Level-0 Automator FM. The operator selects a recovery action in the table at the right-hand side and clicks ‘Recover Run’ to execute it. A run can be started from any state of the subsystems using the ‘Start’ button.

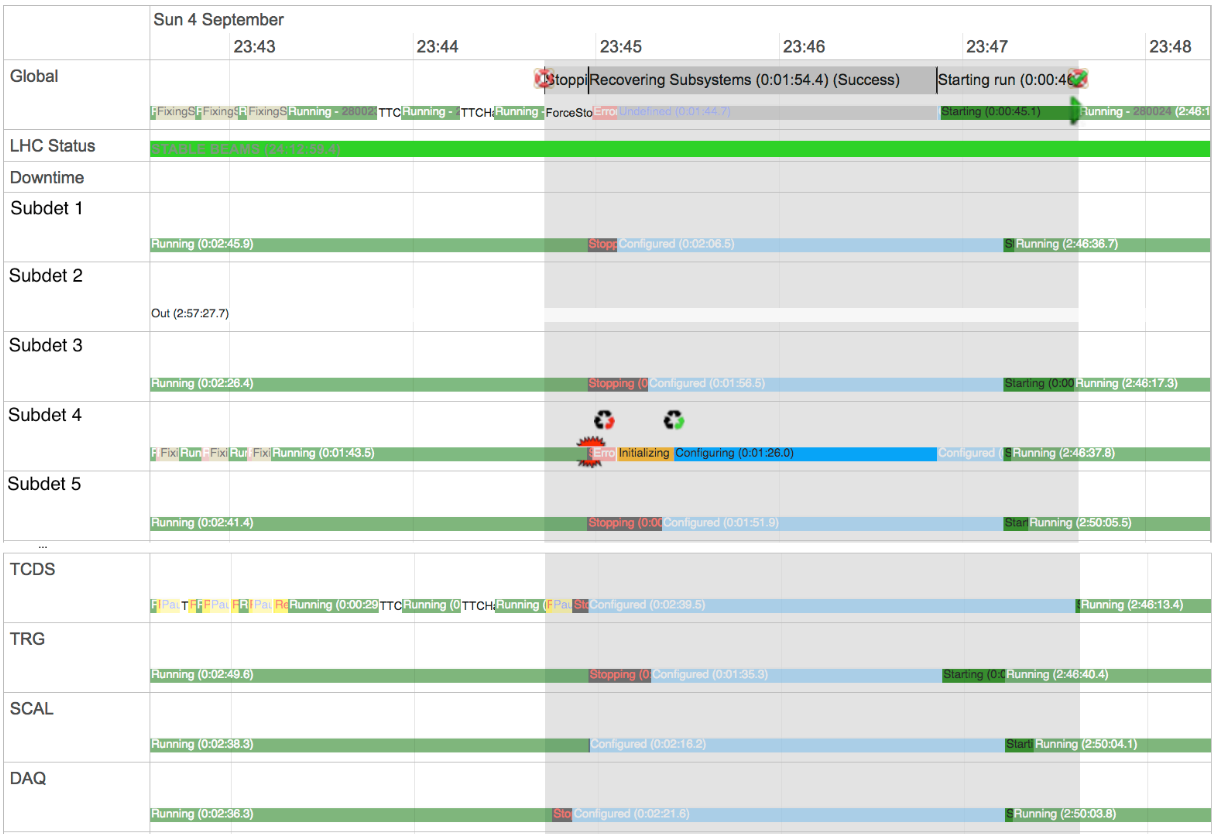


Figure 4. Timeline of a recovery. In this example, around 23:43 subdetector 4 triggers Soft Error Recovery multiple times but this does not seem to recover the problem. Around 23:44, the operator tries to recover by twice issuing a ‘Hard Reset’ command, which also does not seem to fix the problem. The operator then uses the ‘Recover Run’ function of the Level-0 Automator FM, specifying an empty schedule (i.e. no subsystem action between stopping and starting the run). Around 23:45, subdetector 4 fails during the Stopping transition. It is automatically recovered by the Level-0 Automator FM – partly in the shadow of stopping other sub-systems.

Since the operator needs to be informed about subsystem state and configuration and also in some situations still needs to perform manual actions on individual sub-systems, we decided to integrate the user interfaces of the Level-0 Automator and the Level-0 FM. The user interface of the Level-0 Automator is shown as a bar on top of the web page while the Level-0 control page is available below. Whenever the Level-0 Automator executes a recovery or start procedure, the user interface of the Level-0 FM is locked so that the operator cannot interfere by performing manual operations. During other periods, the operator may use the controls of the Level-0 FM, directly.

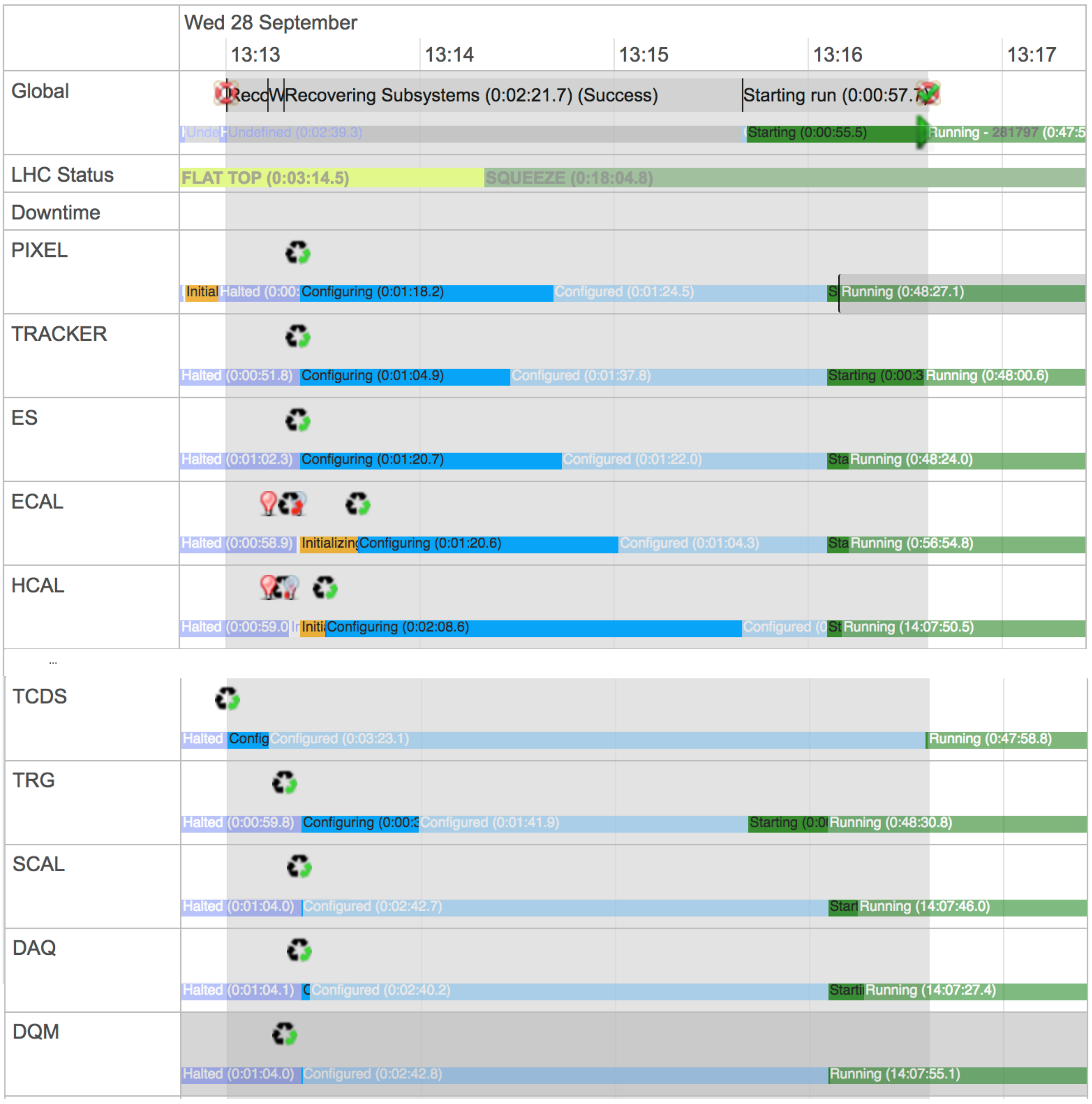


Figure 5. Timeline of starting a run using the 'Start' function of the Level-0 Automator FM. Since the reconfiguration of the TCDS system in this particular case implies a change of clock, several sub-systems require re-initialization in addition to re-configuration.

3.1. Experience

Most DAQ operators immediately embraced the new Level-0 Automator FM and use it to start runs and perform recoveries. Besides knowing the optimal sequence of actions, the Automator also frees the operators from watching the screen to detect completion of steps in a sequence only to trigger the next step. Operators are thus free to focus on other tasks such as communication with the shift crew or documenting their actions.

4. Conclusion and Outlook

We have reviewed the operator assistance features that have been added to the CMS run-control system since the start of operation of the experiment. In particular we have for the first time presented a recent development, the Level-0 Automator, which can execute complex recovery actions in an automated fashion in the most streamlined way. By following the guidance system it ensures consistent configuration of the experiment. It is also able to recover additional problems occurring during the recovery of an initial problem. The tool has sped up recovery from typical error situation and has considerably simplified the task of the operators. In the near future we are planning to integrate the Level-0 Automator with our new DAQ Expert tool in order to automatically trigger recovery when certain error situations are detected.

References

[1] Gulmini M *et al.* 2003 The run control and monitor system for the CMS experiment, presented at *Int. Conf. Computing High Energy and Nuclear Physics*, La Jolla, CA, March 24-28, 2003

[2] Oh A *et al.* 2008 The run control system of the CMS experiment, *J. Phys.: Conf. Ser.* 119 022010

[3] Petrucci A *et al.* 2007 The run control and monitoring system of the CMS experiment, *PoS ACAT* (2007) 026

[4] Sakulin H *et al.* 2012 First operational experience with a high-energy physics run control system based on web technologies, *IEEE Trans. Nucl. Sci.* 59 4 1597-1604

[5] The CMS Collaboration 1994 *CMS Technical Proposal*, CERN LHCC 94-38

[6] The CMS Collaboration 2008 The CMS experiment at CERN LHC, *JINST* 3 S08004 p. 361

[7] Sakulin H *et al.* 2010, Dynamic configuration of the CMS data acquisition system, *J. Phys.: Conf. Ser.* 219 022003

[8] Gutleber J *et al.* 2010 The CMS data acquisition system software, *J. Phys.: Conf. Ser.* 219 022011

[9] Gomez-Reino R *et al.* 2012 Status of the CMS Detector Control System, *J. Phys.: Conf. Ser. 396 012023*

[10] Sakulin H *et al.* 2014 Automating the CMS DAQ, *J. Phys.: Conf. Ser.* 513 012031