INCREASING AVAILABILITY BY IMPLEMENTING SOFTWARE REDUNDANCY IN THE CMS DETECTOR CONTROL SYSTEM

L. Masetti, A. Andronidis, O. Chaze, C. Deldicque, M. Dobson, A. Dupont, D. Gigi,
F. Glege, J. Hegeman, M. Janulis, R. Jiménez Estupiñán, F. Meijers, E. Meschi,
S. Morovic, C. Nunez-Barranco-Fernandez, L. Orsini, A. Petrucci, A. Racz,
P. Roberts, H. Sakulin, C. Schwick, B. Stieger, S. Zaza, CERN, Geneva, Switzerland
P. Zejdl, CERN, Geneva; Fermilab, Batavia, Illinois, USA
U. Behrens, DESY, Hamburg, Germany, O. Holme, ETH Zurich, Switzerland
J. Andre, R. K. Mommsen, V. O'Dell, Fermilab, Batavia, Illinois, USA, G. Darlea,
G. Gomez-Ceballos, C. Paus, K. Sumorok, J. Veverka, MIT, Cambridge, Massachusetts, USA
S. Erhan, UCLA, Los Angeles, California, USA, J. Branson,
S. Cittolin, A. Holzner, M. Pieri, UCSD, La Jolla, California, USA

Abstract

The Detector Control System (DCS) of the Compact Muon Solenoid (CMS) experiment ran with high availability throughout the first physics data-taking period of the Large Hadron Collider (LHC). This was achieved through the consistent improvement of the control software and the provision of a 24-hour expert on-call service. One remaining potential cause of significant downtime was the failure of the computers hosting the DCS software. To minimize the impact of these failures after the restart of the LHC in 2015, it was decided to implement a redundant software layer for the control system where two computers host each DCS application. By customizing and extending the redundancy concept offered by WinCC Open Architecture (WinCC OA), the CMS DCS can now run in a fully redundant software configuration. The implementation involves one host being active, handling all monitoring and control tasks, with the second host running in a minimally functional, passive configuration. Data from the active host is constantly copied to the passive host to enable a rapid switchover as needed. This paper describes details of the implementation and practical experience of redundancy in the CMS DCS.

ROLE OF THE CMS DETECTOR CONTROL SYSTEM AND EXPERIENCE DURING RUN-I

The DCS of the CMS experiment enables the coherent and safe operation of the detector. It provides the supervision of the experiment, allowing for the operation of the sub-detectors and sub-systems and works in synchronization with the LHC, automatically preparing CMS for data taking whenever LHC is ready to generate collisions. The availability of the DCS is crucial to ensure the operation of the experiment and to maximize data taking time.

The CMS DCS consists of an interconnected federation of independent applications, all of which are based on the WinCC OA control system toolkit from ETM professional. The central CMS DCS team takes responsibility for the infrastructure and platform on which the control applications run. This includes Windows operating system, WinCC OA software and other software dependencies such as the OPC DA servers used to communicate with front-end hardware.

WinCC OA is a modular and distributed software package where the functionality is implemented in independent processes, called *managers*, that communicate via the TCP/IP protocol. The central processing unit is the *event manager* which keeps the current process image in memory and ensures the distribution of data to other managers. The process image is organized in typed, structured process variables, called *data points*. A WinCC OA *project* is a configured set of managers running in a single server, typically connected to a local event manager. A WinCC OA *system* identifies a namespace corresponding to one event manager (or two redundant event managers in case of redundant systems). Many systems can be connected forming a distributed network which enables data exchange between systems.

The CMS DCS ran successfully throughout the first data taking run of the LHC with high availability. However, a potential cause of significant downtime was the failure of server hardware, which would immediately stop the hosted control application. In order to restore full functionality of the CMS control system, the central team would intervene urgently to prepare a new server and target the unavailable application to the new computer. This operation was time consuming (up to 1 hour) and could introduce unexpected problems due to the complexity of the installation process.

In order to mitigate the issue of server failure, it was decided to implement redundancy at the control software layer by running each CMS WinCC OA system on two computers, one located in the underground cavern and the other in the computer centre on the surface, in order to overcome a failure of the services or infrastructure at a single geographical location. In this way, if a single server would fail, all CMS DCS control applications would continue to run and a deferred intervention could then be planned to replace the failed hardware. Details on the selected server hardware can be found in [1]. The present paper presents the final software solutions adopted during the full migration of the CMS DCS to the redundant architecture.

ISBN 978-3-95450-148-9

REDUNDANCY IN WINCC OA

The WinCC OA control system toolkit provides a built-in mechanism for achieving high availability through redundant systems. In this approach, two complete, identical instances of each project run on two separate computers, called *peers*. One project runs as the *active* peer while the other runs as a hot standby or *passive* peer. While the system runs in a fully redundant configuration, data changes in the active peer are constantly synchronized across to the passive peer so that it is always up-to-date. If the active peer fails or a manual switchover is requested, the passive peer immediately takes over the role of the active peer.

To avoid both peers sending and receiving value updates from the front-end hardware, the standard WinCC OA drivers running on the passive peer are programmed to discard commands and data point updates.

CONSTRAINTS IN CMS DCS

Although the CMS DCS is based on the WinCC OA toolkit, some implementation decisions and technologies that are used mean that the standard approach of running two identical peers is not feasible or possible. For instance, the CMS DCS makes use of significant quantity of CERN software that has been built upon WinCC OA, such as the JCOP Framework [2]. This software was not initially written with the requirement of supporting redundant systems. While many JCOP components could be quickly modified to become fully compatible with redundant contexts, there were some components where the fundamental design and implementation choices prevented any simple route to achieve redundancy readiness.

Moreover, many hardware devices (PLCs, power supplies) used in CMS do not support connections from two servers, or their performance drops significantly with multiple connections. Also, contrary to typical industrial applications where control systems operate with the same configuration for months or even years, the CMS DCS systems frequently evolve with time. Smooth handling of component installation is essential, so the use of redundancy must not complicate installation for the sub-detector developers. Interruption of the system during upgrades can be tolerated because upgrades are initiated by sub-detector experts who can schedule the intervention at an appropriate time.

An additional factor to be considered is that the behaviour of any non-trivial code running simultaneously on two peers can diverge because of timing issues or the limited functionality of the passive peer. For instance, even if the actions on the process variables (data points) are disabled on the passive peer, all the side effects of the managers, such as sending emails or opening a TCP socket must be disabled in the passive mode otherwise they will be executed twice (once per each peer). To inspect and modify all the potential issues in the code of every CMS sub-detector was not feasible, hence an alternative solution had to be found.

DUPLICATION OF ACTIVE/PASSIVE STATUS

The selected approach to address the aforementioned issues is to run the complete application including drivers and custom scripts only on the active peer and to run a minimal configuration on the passive peer (Fig. 1). This minimal configuration is similar for any application, consisting of the basic WinCC OA managers plus two CMS managers to handle redundancy. Mock drivers, called *simulation managers*, must also exist on the passive side for each corresponding driver on the active side to enable synchronization of the complete set of WinCC OA configuration data.

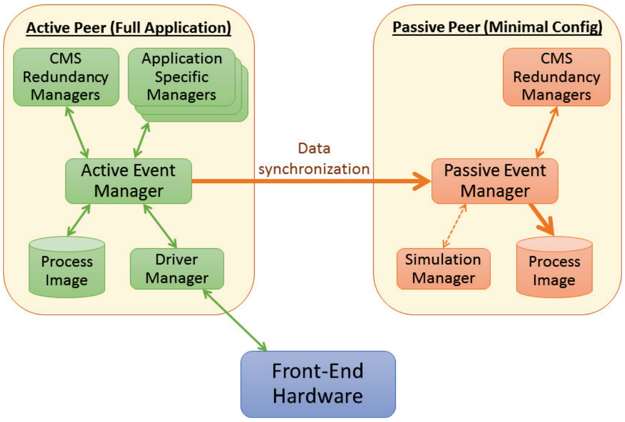
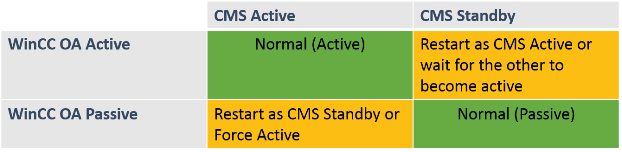


Figure 1: A typical configuration of active and passive peers.

The standard WinCC OA active/passive status of each peer is combined with a CMS-specific active/standby status. WinCC OA status is driven by WinCC OA, depending on the built-in redundancy mechanism. The CMS status reflects the configuration of the running managers: the passive configuration corresponds to the minimal configuration that should run in the passive peer, while any other configuration is considered active. All four combinations of states are possible but only two (WinCC OA Active/CMS Active and WinCC OA Passive/CMS Standby) are stable (Fig. 2).



	CMS Active	CMS Standby
WinCC OA Active	Normal (Active)	Restart as CMS Active or wait for the other to become active
WinCC OA Passive	Restart as CMS Standby or Force Active	Normal (Passive)

Figure 2: Combinations of WinCC OA/CMS Redundancy Status and possible actions to exit unstable configurations.

When an unstable combination is reached, custom logic will act in order to bring the peer to a stable configuration. The action taken depends on several conditions (e.g. checking if the other host is pingable). When a peer detects the configuration WinCC OA Passive/CMS Active, it can exit this situation by forcing itself to be active or by restarting itself with the minimal configuration. A peer that detects the situation WinCC OA Active/ CMS Standby will either perform the transition to become CMS Active or wait for the other peer to become fully active.

INSTALLATION OF COMPONENTS IN A REDUNDANT SYSTEM

All the control software used in the CMS control system is provided in the form of packages, which can be deployed and updated via a dedicated web interface. As the CMS DCS installations are applied to running systems, frequently the start of the installation process involves switching to a safe, reduced functionality state (e.g. stopping the connection to hardware). The installation and upgrade of the components is handled in a dedicated tool developed centrally at CERN in the JCOP framework. This tool supports redundancy and keeps track of which version of the component is installed in each of the two peers. Following this approach the complete installation has to be repeated in each peer. The installation tool provides two operation modes for redundancy:

- *Installation in Active Mode*: The component is installed in the active peer and no redundancy switch is performed.
- *Installation in Split Mode*: When a component must be installed, the system is set to *split mode*, that is the two peers become independent, and the installation is performed on the passive peer. After installation, the passive peer becomes active and redundancy is re-enabled. The process is repeated in the newly passive host and finally the redundancy is enabled again.

However both approaches are problematic in the CMS environment. Using the first approach, any new component will not be installed in the passive peer until it switches to active. This is the wrong moment to install a component, because it will cause unscheduled downtime in the active peer during the critical moment of the switchover. Moreover, in this case, the installation will be unattended because there is no expert explicitly requesting it.

The second approach causes two redundancy switches during the installation of any component. This increases downtime when upgrading/installing a component, which would be tolerable for large, complex installations. For small, rapid upgrades and patches, the overhead of two redundancy switches is less acceptable.

The CMS solution is to install the components in the active peer (as it was done before redundancy was introduced) and then perform a light version of the installation in the passive peer. The possible elements of a typical installation were analysed to find the best way to deploy them to both peers.

- *Changes in the process image (data points, data point types etc.)*: the creation / modification of new data points or data point types is the basic purpose of an installation and does not pose any problem to redundancy because the process image is automatically synchronized by WinCC OA in the passive peer.
- *Configuration files*: Some project configuration files might be updated during installation. These files can be synchronized from the active to the passive peer by the CMS redundancy managers.
- *Addition of new WinCC OA managers*: During installation, new managers might be added to the project in order to perform specific tasks. These managers are not immediately activated in the passive peer in order to preserve the minimally active state, but their configuration is stored internally so that they can be started in case of a switch to CMS active mode.
- *Files inside the project directory*: Local files used in the project are synchronized from the active to the passive peer by the CMS redundancy managers.
- *Installation tool receipt to track which component is installed in each peer*: The receipt can be written on the active peer and automatically synchronized to the passive peer, so that the installation tool believes the component is already installed in the passive peer and will not trigger another installation in case of a switchover.
- *Special actions interacting with the operating system*: The developer can define special actions that interact with the operating system (e.g. modifying the Windows registry to configure an OPC Server). Such actions need to be repeated in both peers and must be placed in special scripts that are executed during the switchover to CMS active mode. The execution of these special actions is delayed until the switchover to ensure that they run with the most recent configuration data.

In case of a redundancy switch, the passive system is activated, already having the latest components installed, and becomes fully operational within a few seconds or several minutes, depending on system conditions and the specific application complexity.

This approach for redundant installation is almost transparent for the sub-detector developer, requiring only minor component changes related to special operating system actions. The central CMS DCS team have minimized these changes by preparing scripts for the most common cases.

REDUNDANCY SWITCHOVER CONDITIONS

The main purpose of redundancy in CMS is to handle the situation where the server hosting a peer fails completely. Further improvements can be made to trigger a switchover in other common failure modes.

Typical checks included in the list of switching conditions are related to the server resources. For example free memory and free disk space are evaluated in each server and cause an increase in the error level of a peer when falling under a certain threshold. This is a default feature of WinCC OA that has been re-used, but the thresholds have been redefined

ISBN 978-3-95450-148-9

because of the large amount of memory available on the servers where the DCS applications run.

Moreover, other application specific switching conditions are configurable in the CMS DCS implementation. The same condition (e.g. is the PLC pingable) is periodically evaluated in two peers. The state is propagated when the condition is stable, i.e. a waiting time (> polling time) has passed since both peers evaluated the condition. This avoids unnecessary switches in case the error is present on both peers, but was detected at different times due to desynchronized polling cycles (e.g. if an essential PLC is not pingable from both peers there is no benefit to switch).

This feature was implemented using *CMSfwClass* [3], a CMS library that implements object-oriented behaviour in WinCC OA, making it straightforward to extend a base class to define new custom conditions.

HANDLING EXTERNAL PROCESSES

The CMS DCS is mainly implemented using WinCC OA but it also needs to run other external processes. These processes typically need to be stopped on the passive peer and started in the active peer following the redundancy switch of WinCC OA. Examples of these processes include the name servers for the custom CERN Data Interchange Protocol (DIP) and Distributed Information Management System (DIM) protocols and OPC servers (that are automatically started with the WinCC OA OPC client but need to be stopped in the passive peer). A centrally-developed CMS tool allows component developers to configure the non-WinCC OA processes to be started or stopped in case of redundancy switch. This mechanism is connected to the CMS active/standby state rather than to the WinCC OA state. This defers the starting/stopping of the process until the complete activation of the project.

SPLIT BRAIN SCENARIO

When the network between two redundant peers goes down, both peers activate because they cannot contact the other server. When the network recovers, one peer must be killed and restarted in passive mode. By WinCC OA default behaviour, peer 2 will always be killed after a network outage. This is not the desired behaviour. Rather we want, if all other conditions are the same in the two peers, to retain the previously active peer in active mode. It was possible to obtain this behaviour by raising the error level of the newly active peer for a few seconds when switching from passive to active state because the other peer is not contactable. By enabling the relevant WinCC OA option, when the two peers reconnect, the one that had the highest error level while disconnected is restarted in passive mode.

AUTOMATIC RESOLUTION OF INTERNAL DATA POINTS

Data points that represent values specific to each individual peer (e.g. manager status, monitoring of the memory or of the disk) need to be duplicated (with the appending of a _2 postfix for the second peer) and configured to be writable also from the passive peer. The CMS DCS team developed an abstraction layer in form of a library that provides redundant compatible versions of the basic WinCC OA functions dpGet, dpSet and dpConnect (used for reading, writing or connecting to a value). These functions will automatically replace the data point names, adding the _2 postfix when needed. The default is to use the data point related to the active peer in a UI connected to both peers and the data point related to the connected peer in case of scripts connected to one peer only. The only modification for the sub-detector developers was to replace the standard WinCC OA functions with the CMS redundant-compatible versions.

CONCLUSIONS

The strategy for redundancy presented in this paper has been successfully applied to the CMS DCS during the first LHC long shutdown and is currently operating in 34 WinCC OA systems running on 29 pairs of servers.

The approach adopted in CMS DCS eliminated the need for adaptations in many third-party software components while enabling the full benefit of an automated switchover to a hot standby system. In this way the CMS DCS does not require urgent expert intervention as the result of issues with a single server.

The implementation fulfils the original requirement of providing robustness against total server hardware failure and, furthermore, triggers a switchover in other failure conditions such as exhaustion of memory and disk space.

By creating an abstraction layer over the redundancy tools of WinCC OA and CMS, the central CMS DCS team minimized the impact on the sub-detector developers in terms of the code changes needed to prepare their applications for running in the redundant environment.

All major issues related to the redundancy were successfully addressed, while keeping additional complexity to a minimum. The current configuration is expected to guarantee high availability and reliability of CMS DCS during the Run-2 phase of the LHC accelerator.

ACKNOWLEDGEMENTS

This work was supported in part by the Swiss National Science Foundation.

REFERENCES

[1] G. Polese et al. “High availability through full redundancy of the CMS detector controls system”, Journal of Physics: Conference Series Volume 396 Part 1 (2012).

[2] O. Holme et al. “The JCOP Framework”, ICALEPCS’05, Geneva, Switzerland, October 2005, WE2.1-6O.

[3] R. Jiménez Estupiñán et al. “Enhancing the Detector Control System of the CMS Experiment with Object Oriented Modelling”, MOPGF025, these proceedings, ICALEPCS’15, Melbourne, Australia (2015).