AUTOMATED COVERAGE TESTER FOR THE ORACLE ARCHIVER OF WINCC OA

A.Voitier, P. Golonka, M. Gonzalez-Berges, CERN, Geneva, Switzerland

Abstract

A large number of control systems at CERN are built with the commercial SCADA tool WinCC OA (formerly PVSS) [1]. They cover projects in the experiments, accelerators and infrastructure. An important component is the Oracle archiver used for long term storage of process data (events) and alarms. The archived data provide feedback to the operators and experts about how the system was behaving at particular moment in the past. In addition a subset of these data is used for offline physics analysis (conditions data). Large volumes of data are produced by the different facilities at CERN (several Terabytes per year). The consistency of the archived data has to be ensured from writing to reading as well as throughout updates of the control systems. The complexity of the archiving subsystem comes from the multiplicity of data types, required performance and other factors such as operating system, environment variables or versions of the different software components. Therefore an automatic tester has been implemented to systematically execute test scenarios under different conditions. The tests are based on scripts which are automatically generated from templates, therefore they can cover a wide range of software contexts. The tester has been fully written in the same software environment as the targeted SCADA system. The current implementation is able to handle over 300 test cases, both for events and alarms. It has enabled to report issues to the provider of WinCC OA. The template mechanism allows sufficient flexibility to adapt the suite of tests to future needs. The developed tools are generic enough to be used to test other parts of the control systems.

INTRODUCTION

In a control system, archiving is responsible for recording the process data and alarms, and then making it available to understand the state a system was in at a certain point in time. The accuracy of the stored data in both value and time is a key aspect for the quality of service offered by an archiver. Operator tools such as trends or reports showing historical values rely on an efficient and accurate archiving system. Misbehaviours of these tools during a data taking period are not acceptable as they could lead to lowered operation capabilities, data loss or even a stop of the concerned system. In addition, upgrades of software components may only be applied at certain periods of time.

A particular requirement of the control systems for the experiments at CERN is to make use of subsets of data stored by the archiver as so called conditions data. The availability and consistency of such data, being for instance the monitored voltages of a high voltage power supply, are then essential and necessary for the off-line physics data reconstruction and analysis. In this example, archived data will act as an input parameter to calculate the actual gain factor of a photomultiplier, which may be linked directly to the efficiency or calibration factors of the detector.

To minimize the possibility of the down-time caused by a control system built with WinCC OA, the different components need to be validated by tests performed in laboratory conditions. This validation is handled by the CERN Industrial Controls group. The group provides a centralized service and support for common controls tools, and performs the initial validation of every new version of the SCADA software components. This include testing their proper operation in new environments (new versions of operating systems, related software, etc.).

Being the least mature and one of the most complex components of WinCC OA, the Oracle archiver requires particular attention. Not only do its scalability and throughput need to be verified (as previously reported in [2]), but also a systematic verification of consistency of archiving data across all possible data types and/or software context is required. The tool presented in this paper is aimed at this kind of coverage testing.

In the paper we will describe the Oracle Archiver, the problems associated with its testing and the approach we have chosen for the architecture of the tool. We will follow by describing the architecture of individual tests, and finally present conclusions.

The tools described in this paper are aimed at testing the behaviour of the Oracle archiver and to check on the integrity of data of various types, in multiple software environments.

To cover the large number of combinations of setups/environments that require testing, a program was devised to automate the – often tedious and complicated – process of setting up the archiving test setup in laboratory conditions. It became possible to repeatedly trigger the tests in the various environments and observe the impact of single environment changes on the result. This part of the project became an independent tool (known as Bootstrapper) that found many other applications.

The testing framework had to be generic and adaptive to cover a wide range of tests. The number of similar test scenarios called for an abstraction: the scenarios can be grouped in families of tests that differ slightly from one another. A powerful mechanism combines templates with test specific configurations to produce the test case scripts. By using 8 templates and some minimal configuration, the current 300 scripts were produced.

Eventually, each issue discovered by the tester would have to be investigated, isolated and reported to the producer of WinCC OA. We came up with an architecture where every test case is self-contained and could eventually be executed with no additional instrumentation, and hence easy to reproduce in a clean laboratory setup.

Software Contexts

The tester is supposed to be repeatedly executed in varying setups (software contexts). It shall be possible to verify the proper functioning of the RDB Archiver in the setups that have different versions and patch-levels of WinCC OA software, Operating System (versions of Linux and Windows supported at CERN), Oracle software, or any other software component that may affect the functioning of RDB Archiver in the future. Other factors such as environment variable settings (e.g. the formatting of time representation, related to the local language setting), security settings, Oracle-specific settings or the Archiver's own settings affect the behaviour of the RDB Archiver and hence extend the spectrum of software contexts in which the tester need to be executed.

Context and Behaviour of the Archiver

The process of storing data is initiated asynchronously by the change of a value (that was declared as having archiving activated). The archiver employs a sophisticated buffering mechanism to optimize the performance with respect to the database, and groups the data before flushing them. As a result, the archived data is available in the database only after a certain "dead time".

The queries for historical data in WinCC OA are performed from scripts, by means of three functions: *dpQuery()*, a versatile SQL-like interface retrieving data, *dpGetPeriod()*, a more restricted interface, and *alertGetPeriod(),* oriented to alarms. The data for historical trend plots come internally through the mechanism used by the *dpGetPeriod()* function. Hence, for the purpose of testing, the functions cover also the trending functionality.

The archiver allows archiving values for the following WinCC OA basic data types: *char, uint, int, float, bool, bit32* (bit field)*, string* and *time*. It also allows archiving array types made of the above basic types (*dyn_int, dyn_string*, etc.).

In addition to the value and time-stamp, the archiver also stores meta-data, such as the status bit fields, defining for instance the quality of the data. Such information can be used later in trend plots to visually mark invalid data.

An outstanding feature of WinCC OA is its very complex and flexible model for alarms. The archiver allows for the storing and querying of complete information about the evolution of alarms in the system. The alarm information is independent from the values, and it changes dynamically as the alarms come, go, are acknowledged and commented. The whole life-cycle of an alarm is traced by the archiver, and this domain also needs to be tested.

The Oracle archiver of WinCC OA has other functionality such as filtering, smoothing (reducing the data), correcting or compressing of archived values. These features have a direct influence on the archiving process. The archiver also allows for database specific maintenance operations such as grouping values in different tables, or taking a range of unused data off-line. These are not covered by the tester at present.

APPROACH AND ARCHITECTURE

The tester was developed iteratively following a progressive build-up of knowledge of the archiver's behaviour. It allowed us to focus the tests on the most critical points. A typical testing scenario was identified as a sequence of actions: trigger the storage of a value (or alert), then retrieve it from the archive, and finally compare the retrieved data with the expected (original), considering values, timestamps and meta-data (status bits).

As already mentioned in the previous chapter the individual test cases are embedded into individual test scripts coded in WinCC OA's own "Control" scripting language (CTRL). Each test script executes the above mentioned sequence of actions and reports the results.

To cover all possible combinations of standard test scenarios for all the WinCC OA data types, the scripts are generated from templates, based on configuration data.

For the automated tester executing numerous test cases in an already pre-configured environment we therefore established the following standard procedure:

1. Retrieve the configuration of the test case.
2. Generate the test script.
3. Execute the script.
4. Visually report and save the result of the test.

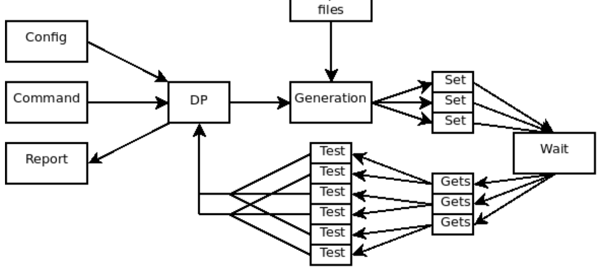


Figure 1: The tester design. (DP=Datapoint).

For all the operations of configuration, command and reporting (shown in Figure 1), the tester relies on WinCC

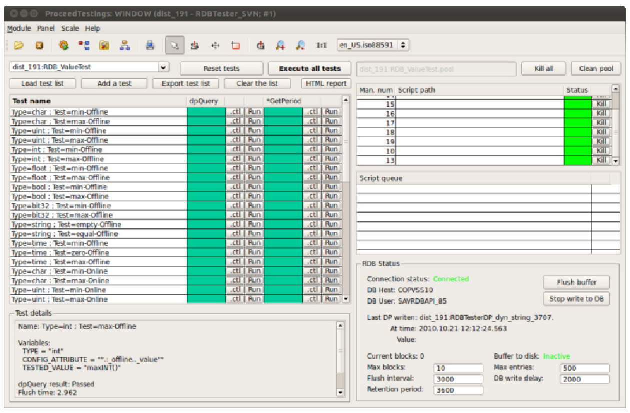
OA datapoints. In addition to being used to link to hardware and process variables, the WinCC OA datapoints represent variables with data-persistency and hence provide for easy inter-processes message passing.

Script Generation

The test scripts are generated whenever needed based on template scripts and adapted to each case through pre-registered configurations. Template files contain the basic structure needed to implement a test of a specific kind. Separate templates are required to address the differences between families of tests are instrumented for the particular data type (basic types, arrays, status bit or alerts) and the retrieval function they use (*dpQuery()* or *dpGetPeriod()*). The templates contain place-holders used during the generation process where they are adapted to the configuration of a test case. A configuration specifies the template file used, the name of the test, its subfamily (e.g. the retrieval function used here for display purposes), and a list of place-holder tags used within the template file. Typically, the place-holder tags allow specification of the data type of the test (*int, char, dyn_float*, etc.), the attribute of the datapoint the test targets (offline value, online value or original value), and the value used for this test.

Parallel Execution System

With a growing number of test cases the need for parallel execution became apparent. WinCC OA offers threading functionality inside the CTRL scripting engine. Due to current limitations these could not be used, therefore, a custom parallel-execution and job-queuing system free of these limitations was developed. The "script pooler" component employs a "pool" of individual CTRL language interpreter processes (managers) to schedule the execution of the subsequent test-case scripts. Each CTRL manager runs in a separate operating system process and is capable of communicating through datapoints, which allowed implementation of the parallel job queuing system. A component provides a panel to deploy and configure a list of CTRL managers for the pool. When deploying managers it is possible to configure the number of managers and pre-allocate them to individual test cases.



Test name	dpQuery			*GetPeriod		
Type=char ; Test=min-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=char ; Test=max-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=uint ; Test=min-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=uint ; Test=max-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=int ; Test=min-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=int ; Test=max-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=float ; Test=min-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=float ; Test=max-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=bool ; Test=min-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=bool ; Test=max-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=bit32 ; Test=min-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=bit32 ; Test=max-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=string ; Test=empty-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=string ; Test=equal-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=time ; Test=min-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=time ; Test=zero-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=time ; Test=max-Offline	🟩	.ctl	Run	🟩	.ctl	Run
Type=char ; Test=min-Online	🟩	.ctl	Run	🟩	.ctl	Run
Type=char ; Test=max-Online	🟩	.ctl	Run	🟩	.ctl	Run
Type=uint ; Test=min-Online	🟩	.ctl	Run	🟩	.ctl	Run
Type=uint ; Test=max-Online	🟩	.ctl	Run	🟩	.ctl	Run

Man. num	Script path	Status	
15		🟩	Kill
16		🟩	Kill
17		🟩	Kill
18		🟩	Kill
19		🟩	Kill
10		🟩	Kill
13		🟩	Kill

Script queue

Test details

```
Name: Type=int ; Test=max-Offline

Variables:
  TYPE = "int"
  CONFIG_ATTRIBUTE = "".:_offline.._value""
  TESTED_VALUE = "maxINT()"

dpQuery result: Passed
Flush time: 2.962
```

RDB Status

```
Connection status: Connected
DB Host: COPVSS10
DB User: SAVRDBAPI_85

Last DP writen: dist_191:RDBTesterDP_dyn_string_3707.
        At time: 2010.10.21 12:12:24.563
          Value:

Current blocks: 0            Buffer to disk: Inactive
Max blocks:        10        Max entries:      500
Flush interval:    3000      DB write delay:   2000
Retention period:  3600
```

Figure 2: The overview panel of the tester.

Reporting

Once a test case has been executed, its script stores the result in a datapoint. It is then accessible later on, which allows running the tester overnight and checking the results later. The information that is stored includes: if the test passed or not, if it raised a warning, the database-flush time in case of success or the error reason otherwise. To display the results, an overview panel (see Figure 2) shows all of the relevant information for the progress of a test suite (all the tests of the same kind are executed in sequence). The panel also shows the script pooler (queue status, managers on duty, commands) as well as summary of the archiver status (latest datapoint written, buffer usage, connectivity, etc.).

TEST DESCRIPTIONS

Each test of the archiver is essentially made of a *Set* operation followed by a corresponding *Get* operation. The result of the *Get* being compared to what has been *Set* previously. Between the *Set* and the *Get* steps, however, the test has to wait for the data to go through the archiver's internal buffers and to get flushed to the database. Thus, we refer to this recurrent procedure as *Set - Wait flush - Get - Test.* Before being applied, this procedure needs to create a randomly named datapoint and configure it to be archived. The *Set* step is performed such that the timestamp on the archived data change is known.

The *Wait* step needs to be synchronized between the tests to reconcile with the buffering behaviour of the archiver. To compensate for a missing functionality within the archiver, a mechanism was implemented to confirm and notify when the archived data reached the database, or that a timeout expires. The *Get* step will then retrieve the previously archived data, using the WinCC OA data-query functions, and extracts the details necessary for the *Test* step. This last step compares the timestamps, the values, the types and the status bits.

For the eight scalar data types, each type is tested in a separate test case. Each test case performs a number of sub-tests, e.g. for successive data point elements (the offline, online and original values), and data-retrieval functions (*dpQuery()* and *dpGetPeriod()*). Finally, several value variations are introduced (e.g. maximum/minimum value for a particular type, positive/negative values). It would be excessively large and not meaningful to test every single possible value for each data type. Therefore only the *min* and *max* functions for numerical types, and an empty and a random value for the string type are tested.

For arrays the tested variations are similar to the scalar values. The special case of an empty array is handled. The size of an array, the type of its elements and the order of the elements are checked.

For the test-case of status bits the original status set is compared to the one retrieved; the actual value and the data type variation are not the subject of the test. Different realistic status values as well as different archiving contexts such as smoothing (data reduction) are set and tested.

For alerts the tests cover the retrieval functions and datapoint attributes, and also the alert configuration itself (number of alarm ranges crossed, type of alarm (value range, discrete value, status bits)). Several scenarios were implemented for alert evolution, with permutations of the *Came, Went* and *Acknowledge* events in the alert life-cycle. Although alerts can display a hysteresis effect because of this sequence of events, the creation of a random test datapoint at the beginning of each test ensures that there is no perturbation between concurrently running alert tests.

RESULTS

An issue with flushing of the archiver's buffer was identified and isolated as soon as we began production testing. Thanks to the tester, the issue which was previously only seen sporadically in production systems could be studied, reproduced and reported to the WinCC OA vendor, and resulted in a fix being made available for CERN users.

The tester identified seven problems with data consistency for scalar data types, namely for special cases such as empty strings or ASCII-0 characters, mis-handling of timezone-setup in the operating system, or bit-field incoherence. Thanks to the requirement of the test-cases being self-contained, it was easy to demonstrate the problems to the vendor, and all of the issues were quickly resolved. Since then, the tester has been reporting the status of all tests for scalar values as "green" (passed).

The tests for array-types detected a further nine issues, which – after being studied – will also be reported to the WinCC OA vendor. Again, problems come up in the handling of non-standard cases such as empty arrays or arrays containing empty strings.

The tests for the status bits did not detect any problems. Nevertheless, the tests reassured us that the problems encountered in previous versions of WinCC OA have not re-appeared. The tests for alerts have not detected any inconsistency so far.

The tester is now routinely used to check all new releases (patches) of WinCC OA that affect the archiver subsystem. Reliable and easy-to-deploy tests significantly reduce the workload on the WinCC OA support team at CERN.

DISCUSSION

The automated coverage testing tool allowed to detect, reproduce, isolate and report a number of issues with the WinCC OA Oracle Archiver, and have them resolved in time for the 2009 run period of the LHC. Similarly, it allowed sorting out an issue with the buffer-flushing mechanism, which affected to the operation of some control systems already in production.

Test cases for array-types have been much more challenging. Firstly, they are harder to debug considering their size-related and ordered nature. Problematic behaviour sometimes appeared only with a specific order of elements. The archiving of array-typed values is nevertheless a feature rarely-used at CERN. Therefore testing and correction of its mechanism was not top priority. However, the results of failed test cases will be reported to the WinCC OA vendor to be corrected in a later version of their product.

For alerts, we decided to not make any further development of the tester until we establish a more efficient testing structure, as the alert life-cycle scenario is too complex for the current approach.

CONCLUSION

In this paper we presented the automated tester for Oracle Archiver of WinCC OA. The tester proved its usefulness in detecting and isolating problems with the archiver, by systematically testing the archiving functionality for all supported WinCC OA data types. Together with the previously discussed scalability-tests [2], the results provided by the automated coverage tester build up our clients' confidence in the Oracle Archiver.

The issues that were easily discovered by the tester confirmed that the chosen approach was the right one. Furthermore, the technical decision of implementing our own parallel-execution system brought additional advantages: a solution offering native parallelisation of CTRL scripts in WinCC OA, and a way to easily isolate the test cases into self-contained scripts, simplifies reporting the issues. Finally, the "Bootstrapper" used to prepare and start the tester evolved into an independent project and finds application beyond its original scope. We believe that the components of the tester may also be used to leverage the testing of other components of WinCC OA, such as the alert system, or the frameworks used to construct the controls applications at CERN.

REFERENCES

[1] ETM, "ETM professional control GmbH - A Siemens Company - WinCC Open Architecture - SCADA System."; http://www.etm.at.

[2] The High Performance Database Archiver for the LHC Experiment. ICALEPCS07, Knoxville, Tennessee, USA.