Integration and validation testing for PhEDEx, DBS and DAS with the PhEDEx LifeCycle agent

C.Boeser[1], T.Chwalek[1], M.Giffels[2], V.Kuznetsov[3], T.Wildish[4]
[1]Institut für Experimentelle Kernphysik, Karlsruhe, Germany, [2]CERN, Geneva, Switzerland, [3]Cornell University, Ithaca, NY, USA, [4]Princeton University, NJ, USA

The ever-increasing amount of data handled by the CMS dataflow and workflow management tools poses new challenges for cross-validation among different systems within CMS experiment at LHC. To approach this problem we developed an integration test suite based on the LifeCycle agent, a tool originally conceived for stress-testing new releases of PhEDEx, the CMS data-placement tool. The LifeCycle agent provides a framework for customising the test workflow in arbitrary ways, and can scale to levels of activity well beyond those seen in normal running. This means we can run realistic performance tests at scales not likely to be seen by the experiment for some years, or with custom topologies to examine particular situations that may cause concern some time in the future.

The LifeCycle agent has recently been enhanced to become a general purpose integration and validation testing tool for major CMS services (PhEDEx, DBS, DAS). It allows cross-system integration tests of all three components to be performed in controlled environments, without interfering with production services.

In this paper we discuss the design and implementation of the LifeCycle agent. We describe how it is used for small-scale debugging and validation tests, and how we extend that to large-scale tests of whole groups of sub-systems. We show how the LifeCycle agent can emulate the action of operators, physicists, or software agents external to the system under test, and how it can be scaled to large and complex systems.

The LifeCycle Agent

The LifeCycle agent can drive arbitrary *workflows* through their lifecycle, in a co-ordinated manner, in parallel, and with different parameters.

A *workflow* consists of a data-structure with an array of *events* and a set of parameters. The agent processes the event-list sequentially through a series of *event-handlers*, each specific to the event being processed.

The workflow object is passed to event-handlers, so they can inspect the parameters and modify them. In this way, a sequence of actions can build on the output of previous actions.

Workflow *templates* make it easy to instantiate multiple workflows with varying parameters. E.g, injection of data at a different site, or at different rates, or subscribed to different destinations.

The LifeCycle agent uses certain workflow parameters to tell it how to run the workflow – the array of event names and their relative timings, flags to control the periodic re-start of a workflow, and so on. Apart from that, the payload structure is completely free-form; it's the responsibility of the event-handlers to interpret it and to populate it so they can effectively communicate with each other.

Event-handlers are dynamically loaded modules or external scripts. To communication with external scripts the workflow is stored on disk as a JSON object, the name of which is passed to the script. The script unpacks the JSON object and does whatever it needs to with the data, then passes its results back in the same way, writing to an output file that was also specified in its command-line arguments. This allows maximal flexibility for external tools, with minimal coupling between them and the LifeCycle agent itself.

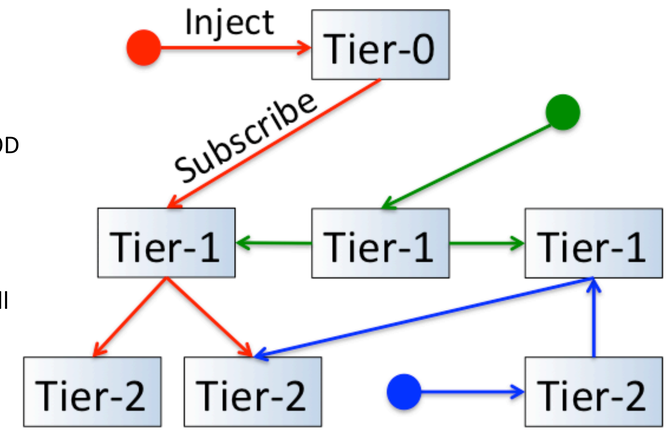
Scale-testing in PhEDEx

The first use of the LifeCycle agent was for PhEDEx scale-testing. Workflows were defined to emulate major data-flows within CMS:

- Inject data at the T0, subscribing that data to a T1, then onwards to one or more T2s
- Inject data at a T2, subscribing it to a T1, then deleting it from the T2 or replicating it to another T2
- Injecti data at a T1 which is then replicated to the other T1s.

By instantiating these templates in different ways, we can emulate the flow of custodial RAW data, of monte-carlo produced at T2's, of AOD production at T1s, and more.

We use fake data-transfers for a realistic emulation of the behaviour of PhEDEx. We can test at scales well beyond what our production infrastructure could handle, which shows us that PhEDEx will scale to our needs over the coming years.



Integration and regression testing in PhEDEx

The LifeCycle agent is used to test PhEDEx releases. We set up a private PhEDEx instance with a few nodes, and drive transfers from one node to another using the LifeCycle agent to orchestrate the flow.

For special cases, such as bugs that occur under unusual conditions, the LifeCycle agent can be configured to reproduce these conditions, repeatedly and systematically.

One of the harder things to test is web-site access by different users with different access-rights. Rather than requiring the support of several people to systematically test a new release of the website, with all the coordination and overhead that implies, we can use the LifeCycle agent again.

We extend the private testbed to include a website, with self-signed certificates allocated to specific roles. The LifeCycle agent uses the website to perform a series of actions (create request, approve it, change its priority), repeating this workflow for each certificate, and for different source and destination nodes. It checks that actions succeed where they should and fail where appropriate.

This gives greater code-coverage than is possible if only one role is used, that of the developer testing the release.

Use in HEPiX IPv6 and ANSE projects

Beyond CMS, The LifeCycle agent is also used by the HEPiX IPv6 and ANSE (Advanced Network Services for Experiments) projects.

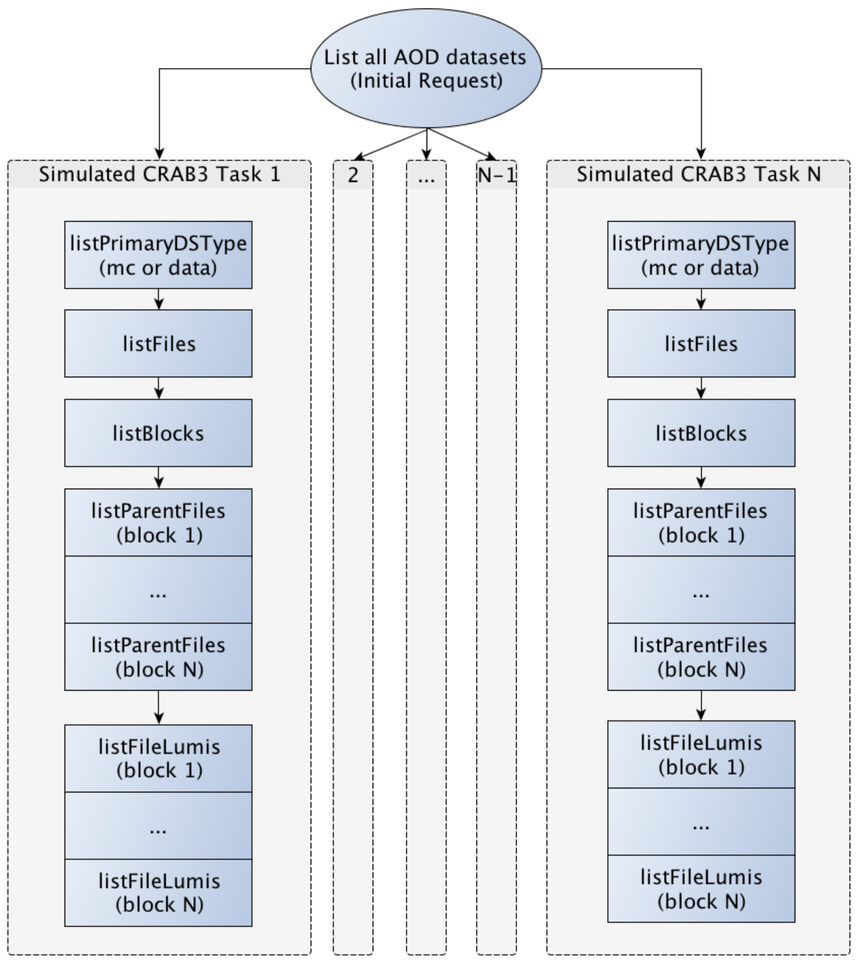
The IPv6 group use it for long-term testing of file transfers on their testbed, to test components of grid and experiment software stacks. ANSE uses it for high-rate transfer testing with a private PhEDEx instance.

Scale testing of DBS3

Since DBS is written in Python, a python framework to facilitate interacting with the LifeCycle agent has been developed. This framework decodes and re-encodes the JSON payload, provides timing and other statistical measures, and simplifies building HTTP API calls from payload objects. The collected data can be sent back to the LifeCycle agent or exported for further analysis in a SQLite database.

A separate package, 'LifeCycle Analysis', was created to analyse the results from the tests. A histogram manager uses ROOT to create timing and error distribution histograms for all the APIs used in the test. This framework is generic enough to be exportable to any python-based tool that wishes to interact with the LifeCycle agent.

Scale tests were performed using separate LifeCycle agents and workflows submitted as batch jobs, which allows an arbitrary number of agents to be run against the server being tested.

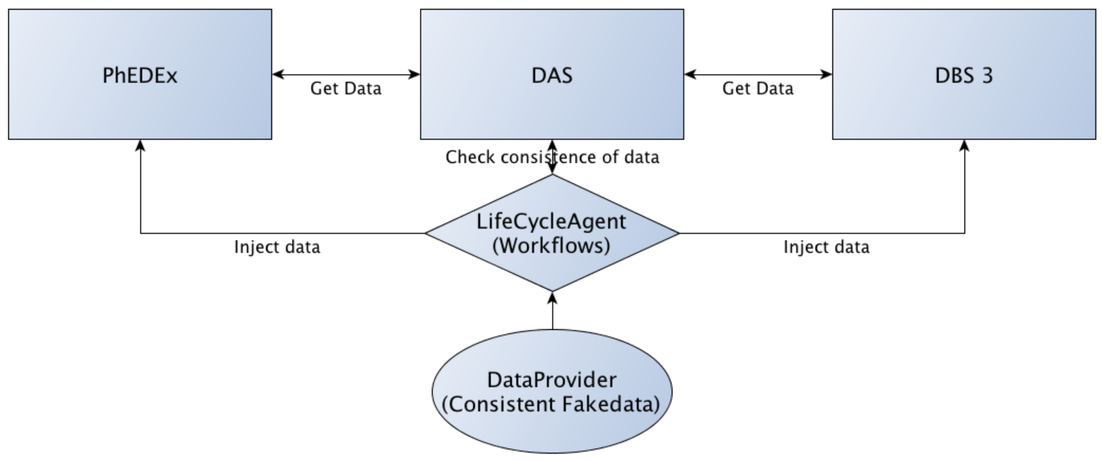


Integration testing with PhEDEx, DAS and DBS

We used the LifeCycle agent for integration-testing of PhEDEx, DAS, and DBS. Fake data was created and injected into PhEDEx and DBS. DAS then retrieved information about the data from both sources, and compared the results. Any mismatches were flagged.

By injecting specific errors in either PhEDEx or DBS (changing filenames etc) we can fake errors that we expect to detect with DAS. Special event-handlers are used to compare the errors detected by DAS with the injected errors, and alert us to any unexpected failures.

This is the first time we have been able to do this sort of integration testing in CMS, with several components of the computing system, and without using production resources.



References

- Integrating the Network into LHC Experiments: Update on the ANSE Project (CHEP 2013)
- WLCG and IPv6 – the HEPiX IPv6 Working Group (CHEP 2013)
- The CMS Data Management System (CHEP 2013)
- Data Bookkeeping Service 3 – Providing event metadata in CMS (CHEP 2013)