The Memory of MICE: The Configuration Database

A J Wilson¹, D J Colling², P Hanlet³

¹Rutherford Appleton Laboratory, Harwell Oxford, Oxon, OX11 0QX, UK
²Imperial College, London, UK
³Illinois Institute of Technology, Illinois, USA

E-mail antony.wilson@stfc.ac.uk

Abstract. The configuration database (CDB) is the memory of the Muon Ionisation Cooling Experiment (MICE). Its principle aim is to store temporal data associated with the running of the experiment; these data are used throughout the life cycle of experiment, from running the experiment through data analysis. The CDB also serves as a moderator in the MICE state machine by defining allowable operating states of subsystems depending on the overall state of MICE and other subsystems. Master and slave CDBs, with multiple mirrored pair raid arrays, have been set up in different parts of the site to increase resilience, as well as off site backups. Access to the CDB is via a Python API, which communicates with a WSDL interface provided by a web-service on the CDB. The priority is to ensure availability of the CDB in the experiment control room. The master CDB is located in the MICE control where it is only used by the running experiment. In the event of the failure of the master, the slave can easily be promoted to master. Read only access to the CDB for data analysis and reconstruction is provided by the slave which has an up to the minute copy of the data. As MICE is a precision experiment which will measure a 10% muon cooling effect with 1% precision, it is imperative that we minimize our systematic errors; the CDB will ensure reproducible and documented running conditions in a highly resilient manner. A description of the hardware and software used in the MICE CDB will be described in what follows.

1. Introduction

The international Muon Ionization Cooling Experiment (MICE) is under development at the Rutherford Appleton Laboratory (RAL), UK. MICE will demonstrate, for the first time, the effect of muon ionization cooling in low-Z absorbers for use in future accelerators for a neutrino factory and/or muon collider. The configuration database (CDB) is the memory of the MICE. Its principle aim is to store temporal data associated with the running conditions of the experiment. These data can change on a per run basis (e.g. magnet currents, high voltages), or on long time scales (e.g. cabling, calibration, and geometry). These data are used throughout the life cycle of experiment, from running the experiment through to data analysis.

The CDB has expanded its role from its initial design of simply storing configuration data, to also forming an essential part of the MICE state machine as used by the controls and monitoring system. It is also used to store and query additional data which includes the settings for alarm limits for use by the controls and monitoring system and the geometry of the experimental apparatus as derived from CAD models.

This paper provides an overview of the software and hardware architecture used to provide a resilient system. The planned use of the CDB within the MICE state machine is also described.
2. Architecture

The use cases for accessing the CDB can be broadly split up into two:

i) Read write access from the control room during the running of the experiment.

ii) Read only access from on and off site by scientists for data analysis.

The priority is to ensure availability of the CDB in the MICE control room when running the experiment. MICE must be able to run even if it is cut off from the rest of the site. An overview of the architecture is given in figure 1.

![Diagram of the configuration database architecture]

**Figure 1.** The components of the configuration database.

2.1. Software

The CDB encompasses a number of components: the database, the web service, the client API. The database used is Postgres, Tomcat is used as the servlet container and the client API is in Python.

2.2. Hardware

Master and slave CDBs have been set up in different parts of the RAL site to increase resilience. Both machines have multiple mirrored pair raid arrays, with the data stored on one mirrored pair and the database transaction logs stored on another mirrored pair of each machine. Off site backups of the data are also kept.
The master CDB is located in the control area where it is only used by the running experiment, see figure 1. In the event of the failure of the master, the slave can be promoted and the control room services can be switched to use the new master in a user transparent way. There is near instantaneous replication of data between machines, which is achieved using the replication functionality introduced with Postgres 9. The web service on the slave is also only accessible from machines in the control area.

Access to the CDB for anyone outside the control area is via an additional front end node that runs a copy of the web service. This web service provides read only access to the CDB slave for scientists doing data analysis, ensuring that user activity cannot impact on the running of the experiment.

3. CDB and the MICE State Machine

The MICE experiment is in the process of being modeled as a set of state machines for use by the controls and monitoring system. The MICE state machine can have states: off, powered, standby, testing, and running. Furthermore, there are a number of subsystems in MICE, each of which have various states. The allowable operational states of any of these subsystems depends on the complex relationships between the present MICE state and other subsystems. The purpose of the MICE state machine to ensure that only compatible states of the various subsystems of MICE can be set. There is a minimal amount of logic in the CDB which uses the data to decide whether a state change is allowed. Decisions are based on the current states and allowed states. The CDB stores the data that describe the current state of the MICE state machine, Table 1 along with data that describe allowed state transitions, Table 2. The state machine is data driven.

### Table 1. The currentstate table

| Column Name       | Description                                      |
|-------------------|--------------------------------------------------|
| System            | The name of the system                           |
| State             | The state of the system                          |
| Valid from time   | The time this state was set                      |
| Valid until time  | The time the state changed to a different state  |

Along with the current states of all of the systems the CDB also holds the history of all state changes, Table 1, namely the “valid from” and “valid until” times. The current value for a system is initialized with “valid until” of null, and only changes when the state of a machine is updated.

### Table 2. The statemachine table.

| Column Name       | Description                                      |
|-------------------|--------------------------------------------------|
| System            | The name of the system                           |
| State             | The state of the system                          |
| Related System    | The name of a related system                     |
| Related System State | An allowed state of the related system          |

3.1. A Simple Example

The functionality of the CDB is best illustrated with a simple example that has two systems, MICE and power supply. The data for this example is shown in Table 3.

International Conference on Computing in High Energy and Nuclear Physics 2012 (CHEP2012) IOP Publishing
Journal of Physics: Conference Series 396 (2012) 052075 doi:10.1088/1742-6596/396/5/052075
Table 3. An example of the contents of the table statemachine.

| System      | State  | Related System | Related System State |
|-------------|--------|----------------|----------------------|
| MICE        | OFF    | MICE           | POWERED              |
| MICE        | POWERED| MICE           | OFF                  |
| MICE        | POWERED| MICE           | STANDBY              |
| MICE        | STANDBY| MICE           | POWERED              |
| MICE        | OFF    | POWER SUPPLY   | OFF                  |
| MICE        | POWERED| POWER SUPPLY   | OFF                  |
| MICE        | POWERED| POWER SUPPLY   | POWERED              |
| MICE        | POWERED| POWER SUPPLY   | CIRCUIT BREAKER CLOSED|
|             | STANDBY| POWER SUPPLY   | CIRCUIT BREAKER CLOSED|
| POWER SUPPLY| OFF    | POWER SUPPLY   | POWERED              |
| POWER SUPPLY| POWERED| POWER SUPPLY   | OFF                  |
| POWER SUPPLY| POWERED| POWER SUPPLY   | CIRCUIT BREAKER CLOSED|
| POWER SUPPLY| CIRCUIT BREAKER CLOSED| POWER SUPPLY   | POWERED              |

Consider the case where both systems have an initial state of OFF as recorded in the currentstate table. If we want to put MICE into the STANDBY state then we must go through these transitions:

- MICE state change OFF -> POWERED
- POWER SUPPLY state change OFF -> POWERED
- POWER SUPPLY state change POWERED -> CIRCUIT BREAKER CLOSED
- MICE state change POWERED -> STANDBY

The only state that MICE can go to from OFF is POWERED, and it cannot progress to STANDBY until the related systems change state; in this case the power supply must have a state of CIRCUIT BREAKER CLOSED.

Going back to the initial states of OFF for both systems again, the use of the CDB will now be described. The CDB can be used to get the current state of a system:

getCurrentState('MICE')
This results in the SQL query:

```
select state from currentstate where system = 'MICE' and validuntiltime = null;
```
This returns ‘OFF’

getCurrentState('POWER SUPPLY')
This results in the SQL query:

```
select state from currentstate where system = 'POWER SUPPLY' and validuntiltime = null;
```
This returns ‘OFF’

The CDB can be used to try and change state:

setState('MICE', 'POWERED')
Before setting the state, check if this transition is allowed. To do this we must find the allowed related states and the current states.

a) Find allowed related states:
This results in the SQL query:

```
select relatedsystem, relatedstate from statemachine where system = 'MICE' and
```
state = ‘POWERED’;
Returns relatedsystem = ‘MICE’, state = ‘OFF’
relatedsystem = ‘POWER SUPPLY’, state = ‘OFF’
relatedsystem = ‘POWER SUPPLY’, state = ‘POWERED’

This is actually getting the list of states from which we can make the transition to the new state i.e. all current states must be contained in this list for the transition to be allowed.

b) Get current state:
This results in the SQL query:
select system, state from currentstate where validuntiltime = null;
Returns system = ‘MICE’, state = ‘OFF’
system = ‘POWER SUPPLY’, state = ‘OFF’

C) Check transition allowed:
Each system’s current state must appear in the results from the first query for the transition to be allowed. For this example this is true so we can update the database. This simple algorithm can handle multiple related systems.

3.2. A More Complex Example
Representing just the MICE and power supply systems requires 65 rows of data, which covers five states of MICE and seven states of power supply. In reality power supply is one of five systems of the decay solenoid, Table 4.

Table 4. A more complex example.

| Decay Solenoid | MICE States       | Off | Powered | Standby | Testing | Running |
|----------------|-------------------|-----|---------|---------|---------|---------|
| Off            |                   | X   | X       |         |         |         |
| Powered        |                   |     |         |         |         |         |
| Standby        |                   |     |         |         |         |         |
| Pumping        |                   |     |         |         |         |         |
| Vacuum Ready   |                   | X   |         |         |         |         |
| Fault/Loss Vacuum |             |     |         |         |         |         |
| Services       |                   |     |         |         |         |         |
| Off            |                   | X   |         |         |         |         |
| Ready          |                   |     |         |         |         |         |
| Linde Refrigerator |              | X   |         |         |         |         |
| Off            |                   | X   |         |         |         |         |
| Powered        |                   |     |         |         |         |         |
| Compressor On  |                   |     |         |         |         |         |
| Cool-down      |                   |     |         |         |         |         |
| Warm-up        |                   |     |         |         |         |         |
| Refrigerator Ready |             | X   |         |         |         |         |
| Fault/Quench   |                   |     |         |         |         |         |
| Cryogenics     |                   | X   |         |         |         |         |
| Off            |                   | X   |         |         |         |         |
| Powered        |                   |     |         |         |         |         |
| Safe Mode      |                   |     |         |         |         |         |
| Manual         |                   |     |         |         |         |         |
| Parallel Cool-down |         | X   |         |         |         |         |
| L-He Level Control |              |     |         |         |         |         |
| Power Supply   |                   | X   |         |         |         |         |
| Off            |                   |     |         |         |         |         |
| Powered        |                   |     |         |         |         |         |
Representing all of these states and transitions requires in depth knowledge of each of the individual systems along with their interdependencies. Table 4 only shows the relationships of the states of the components of the decay solenoid with the MICE state machine. There are also relationships between components of the decay solenoid.

3.3. Conclusions

The CDB software and hardware configuration and deployment has been designed to provide a highly resilient system for the MICE control room systems, while still allowing access to the latest data for scientists on and off site.

Aside from providing storage and retrieval of run time conditions, the CDB provides the MICE state machine. The MICE state machine is potentially a very complex system. It is impossible to remove all complexity; it must be managed in an appropriate way. The code for the MICE state machine in the CDB is relatively simple, so it is easy to debug and maintain. The complexity is the mapping of allowed states and transition, which can be done on a system by system basis helping to reduce complexity. This mapping is best done by the experts of the separate systems with the resultant mappings being stored in a database.

MICE is a precision experiment, it is imperative that we minimize our systematic errors; the CDB will ensure reproducible and documented running conditions in a highly resilient manner. This information is crucial to the running of the experiment and understanding the experimental data.