Conditions Databases: the interfaces between the different ATLAS systems

A.Amorim, D.Klose, L. Pedro, N.Barros, T.Franco
Faculty of Sciences of the University of Lisbon (FCUL), Lisbon, Portugal
D. Burckhart-Chromek, J. Cook, M. Dobson, J. Flammer, R. Hawkings, R. Jones, D. Liko, L. Mapelli
European Organization for Nuclear Research (CERN), Geneva, Switzerland
A. Perus, A. Schaffer, Laboratoire de l'Accelerateur Lineaire (LAL), Orsay, France
D. Malon, Argonne National Laboratory (ANL), Argonne, Illinois, USA
E. Badescu, M. Caprini
National Institute of Physics and Nuclear Engineering (IFIN-HH), Bucharest, Romania
A. Kazarov, I. Soloviev, Y. Ryabov
Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg, Russia
I. Alexandrov, V. Kotov, M. Mineev
Joint Institute for Nuclear Research (JINR), Dubna, Russia
S. Kolos*, University of California, Irvine, USA

Abstract

Conditions Databases are beginning to be widely used in the ATLAS experiment. Conditions data are time-varying data describing the state of the detector used to reconstruct the event data. This includes all sorts of slowly evolving data like detector alignment, calibration, monitoring and data from Detector Control System (DCS).

In this paper we'll present the interfaces between the ConditionsDB and the DCS, Trigger and Data Acquisition (TDAQ) and offline control framework (Athena).

In the DCS case, a PVSS API Manager was developed based on the C++ interface for the ConditionsDB. The Manager links to a selection of datapoints and stores any value change in the ConditionsDB. The structure associated to each datapoint is mapped to a table that reflects this structure and is stored in the database.

The ConditionsDB Interface to the TDAQ (CDI) is a service provided by the Online Software that acts as an intermediary between TDAQ producers and consumers of conditions data. CDI provides the pathway to the ConditionsDB information regarding the present or past condition of the detector and trigger system as well as all the operational and monitoring data. It provides the link between the Information Service (IS) and the ConditionsDB

Conditions database integration into the ATLAS Athena framework is also described, including connections to Athena's transient interval-of-validity management, conversion services to support conditions data I/O into Athena transient stores, and mechanisms by which the conditions database may be used for timestamp-mediated access to data stored in other technologies such as NOVA and POOL.

INTRODUCTION

Conditions Databases [1] are beginning to be widely used in the ATLAS experiment. The Conditions Database contains conditions data, that is, data describing the state of the detector for a given time and that are necessary to reconstruct the event data. Conditions data are typically time-varying data which evolve slowly and include detector alignment data, calibration data, monitoring data and Detector Control System (DCS) data. Since all this data is provided or required by other systems, it is necessary to provide interfaces to each client and provider system. In the following sections we describe the existing interfaces between the conditions database and the DCS, Online Software and ATHENA framework.

THE DCS INTERFACE

The DCS [3] is controlled and managed by the PVSS [5] SCADA (Supervisory Control And Data Aquisition) system. The PVSS structure consists of an Event Manager which is the core manager and of specific managers for each task, such as Data Manager, Control Manager and Archive Manager. There are several other managers provided by PVSS for other tasks such as for hardware control. For specific tasks, for which no manager is provided by PVSS, there is the possibility to create a custom made API Manager which can integrate the PVSS system seamlessly. Since DCS data are conditions data, it was necessary to come up with a way to export this data from PVSS to the Conditions DB. After studying the different possibilities to export data from PVSS, the best solution found, that is, the most efficient and easiest to use, was the creation of a custom API manager developed in C++, based on the C++ interface for the Conditions Database.

The manager requires a special datapoint [1] from which it reads the configuration. This configuration can be done through PVSS using PVSS graphic panels (see figures 1 and 2). One of these panels allows to configure the database connection (host, user, password) and the parent folder

* On leave from PNPI

[1] "datapoints" are special structures in which PVSS stores the data internally

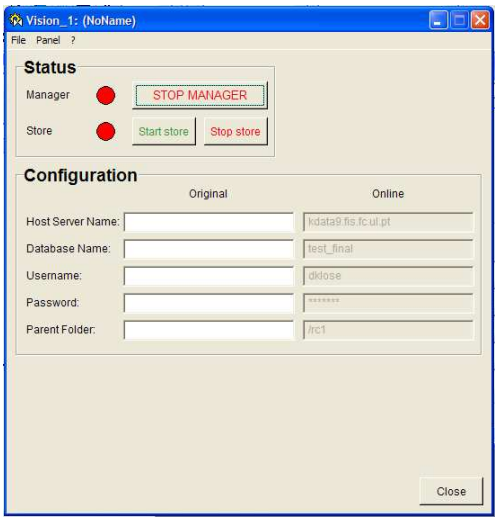

Figure 1: Panel for manager configuration

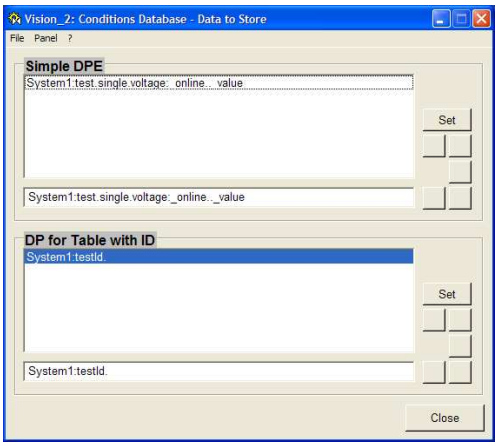

Figure 2: Panel for datapoint selection

(which allows to divide data in a hierarchical way). It also allows to start and stop the manager and data storage. The other panel allows to define the set of datapoints to be stored as structures and the set of datapoints to be stored as structures with channel Id. When running, the API manager connects to each of the defined datapoints and stores the initial value in the database, defining a time interval from since_t to till_t. The since_t is set to the timestamp of the PVSS message and the till_t is set to infinite. When a value change occurs, the time interval of the previous value is updated by setting the till_t of the previous entry to the timestamp of the new PVSS message, and the new value is stored in the database in the same way as the first value. If the datapoint is a structure, this structure is stored in the database as a table which reflects the structure. Supported datapoint types are: int, long, float, double, bool, string and dynamic arrays of each of these types (except bool). In case anything goes wrong with the database connection, or the data storage, the API manager sends the respective error message to the PVSS system. The PVSS API manager is available on the web [6] in a package which includes the binary, the PVSS panels and a quick guide.

THE CONDITIONS DB INTERFACE TO TDAQ

The Conditions Database Interface CDI is a service provided by the Online Software [2] that allows interfacing between the components of the Online software and the ATLAS Conditions Database. The system can be seen as having two main objectives:

- to act as an intermediary between some TDAQ producers and consumers of conditions data, the goal being to provide the interested systems information regarding the present or past condition of the detector and trigger system that is required for their correct operation;
- to provide a pathway between the TDAQ system and the Conditions Database repository that will allow TDAQ clients to register permanently conditions information that is required for later analysis of raw data.

In its current implementation, the CDI establishes an interface to the Online Software Information Service (IS), providing all the necessary operational mechanisms that allow data retrieval from IS and storage in the ConditionsDB (see figure 3. This is done by implementing the functionality to store data published in subscribed IS servers into the conditions database. The subscription method from the CDI is based on a quasi-dynamic process: the CDI subscribes for the RunParams.Conditions information which contains a multi-string value. This information provides to the CDI the IS data sources that will be used to gather information. If the RunParams.Conditions is updated, the CDI will automatically re-configure itself in order to update the IS data sources.

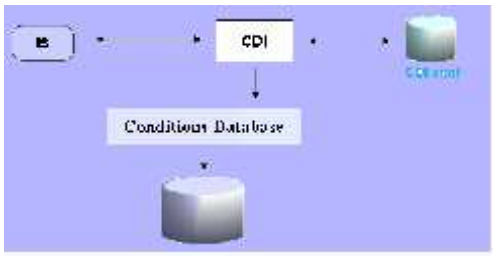

Figure 3: How CDI works

For storage, the CDI creates a CondDBTable for each data source, with the proper structure that is provided by the IS meta-information repository. If this information is not available, a generic CondDBTable with a blob will be created. Each time the *RunParams.Conditions* is updated with a new value, the CDI will re-create the CondDBTables for each IS source and create a folder in the

Table 1: Example of table structure

Name	Type
since_t	bigint(20)
till_t	bigint(20)
run_number	bigint(20)
max_events	bigint(20)
rec_enable	bigint(20)
trigger_type	bigint(20)
detector_mask	bigint(20)
beam_type	bigint(20)
beam_energy	bigint(20)
filename_tag	text

conditions database that reflects the structure from the IS source. When a new value from one (or more) of these IS data sources arrive, the CDI stores a new value in the database in the proper folder. For example the informations from *RunParams.RunParams* will be stored with the following structure in the database in folder */tdaq/partition-name/RunParams.RunParams*:

At storage, the till_t is always infinite, but when an update for each Information occurs, a new entry will be stored in the database and the time interval for the previous entry is closed, i.e. the till_t of the previous entry is set to the since_t of the new entry. Through the CDI all IS data considered to be conditions data is stored in the conditions database.

THE ATHENA INTERFACE

ATHENA is the ATLAS Common Framework. It is based upon the GAUDI framework which was initially developed by LHCb but is now a common project with ATLAS. ATHENA [4] is a part of the ATLAS Offline Computing project and requires access to the conditions database for the offline computing algorithms. Therefore, since ATHENA is a client for conditions data, it became necessary to read the conditions data from ATHENA, especially data stored from DCS and IS. The data access in ATHENA is made in two steps:

- The IOVSvc handles the Interval of Validity (IOV) of the required object asking IOVDbSvc for the address of the object for the valid time.
- The data is retrieved by the Conversion Service and sent to the Transient Detector Store.

When a request is sent to the StoreGate the IOVSvc is responsible to define the IOV of the object requested. In order to do this it uses the IOVDbSvc which connects to the ATLAS Conditions Database and retrieves the IOV of the object and the address to it's data.

In the address fetched by the IOVDbSvc there is information for the Conversion Service that is responsible to get the object data. To retrieve data stored in the ATLAS Conditions Database the responsability is from the CondDB-MySQLCnvSvc (see figure 4).

The IOVDbSvc sends the information about the location of the CondDBtable to the CondDBMySQLCnvSvc. Then the data pointed by the IOVDbSvc is read by the CondDB-MySQLCnvSvc which has also the responsability to copy the data from the Conditions Database CondDBTable to an ATHENA object.

In order to be able to access this data from ATHENA, a new object was created in ATHENA, called GenericDbTable. This object is the equivalent of the CondDBTable used in the Conditions DB API and has all the methods necessary to access the data contained in it. Since this object is polymorphic, it has a very versatile structure which allows to use it for all objects stored in the Conditions Database. in other words the GenericDbTable is a transient table wich has a structure very similar to the persistency table that contains the data.

The GenericDbTable uses the same technology of the CondDBTable having the same methods to retrieve the object structure and its contents. However the GenericDbTable can't handle the interval of validy of the data, unlike the CondDBTable. This comes from the fact that in the ATHENA framework the object IOV and data are handle by different services.

The CondDBMySQLCnvSvc is a conversion service that can link to the Conditios Database and interpret the CondDBTable from the Conditions Database API and store this data to the transient Athena object. This conversion service acts as an intermediate layer between the Persistency and Transient Stores.

Using the ATHENA StoreGate pointers, any Conditions Database data can be read from the ATHENA framework in a way transparent to the user.

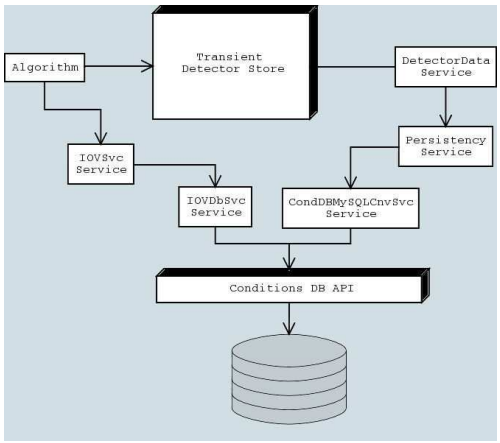

Figure 4: How the Athena interface works

ACKNOWLEDGEMENTS

We thank all of our ATLAS collaborators and, in particular, the developers from the online and offline software

domains. We also thank the users of our software for the valuable feedback that helps us improve it. Our special thanks to R.D. for his valuable help on the development of the CondDBMySQLCnvSvc. Our thanks go also to Serguei, for his help on the development of the CDI and Mihai for the valuable and constructive comments .

REFERENCES

[1] A. Amorim, J. Lima, L. Pedro, D. Klose, C. Oliveira, N. Barros. IEEE-NPSS: An Implementation for the ATLAS Conditions Data Management Based on Relational DBMSs. In Proceedings of the 13th IEEE-NPSS Real Time Conference, pp. 591- 595, May 2003.

[2] ATLAS Online Software
http://cern.ch/atlas-onlsw

[3] ATLAS Detector Control System
http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DCS/dcshome.html

[4] ATLAS Offline Computing
http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/

[5] http://www.pvss.com

[6] http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DCS/CONDDB/CondDBhome.html