# FESA 3.0: OVERCOMING THE XML/RDBMS IMPEDANCE MISMATCH

M. Peryt, M. Martín Márquez, CERN, Geneva, Switzerland

*Abstract*

The Front End System Architecture (FESA) framework developed at CERN takes an XML-centric approach to modelling accelerator equipment software. Among other techniques, XML Schema is used for abstract model validation, while XSLT drives the generation of code. At the same time all the information generated and used by the FESA framework is just a relatively small subset of a much wider realm of Controls Configuration data stored in a dedicated database and represented as a sophisticated relational model. Some data transformations occur in the XML universe, while others are handled by the database, depending on which technology is a better fit for the task at hand. This paper describes our approach to dealing with what we call the "XML/Relational impedance mismatch" – by analogy to Object/Relational impedance mismatch – that is how to best leverage the power of an RDBMS as a back-end for an XML-driven framework. We discuss which techniques work best for us, what to avoid, where the potential pitfalls lie. All this is based on several years of experience with a living system used to control the world's biggest accelerator complex.

## INTRODUCTION

The Front-End Software Architecture (FESA) [1] framework is a comprehensive environment covering all aspects of the development of real-time control software for front-end computers. Through a mix of modelling techniques, automatic code generation and custom code the equipment specialist can quickly develop, test and deploy software for accelerator devices. The *XML* technologies are the cornerstone of FESA approach and are used at every single stage of its workflow [2].

The CERN accelerator complex control systems are fully data-driven thanks to the Controls Configuration Database (CCDB). Its elaborate *relational* schema and supporting tools allow configuring all layers of the accelerator controls: from the top-level graphical tools used by the control room operators, through the middleware down to the hardware drivers and front-end computer start-up sequences.

The FESA project was launched in 2003. Since then it went through many iterations. The current production release is version 2.10, while new major redesign FESA 3.0 is in the works and scheduled for release in late 2009.

The CCDB has a much longer history dating back to the early 1990s. It has gone through countless iterations but has always used Oracle® RDBMS and the suite of tools and technologies provided by this vendor.

The following sections describe the intricate relationships of FESA and CCDB, the challenges posed by the use of XML and relational technology stacks and the solutions employed to make them work together.

## FESA/CCDB WORKFLOW

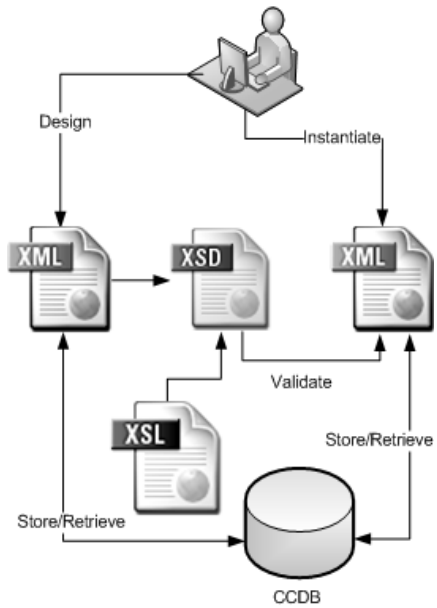


Figure 1: FESA workflow until version 2.10.

When an equipment specialist starts the process of modelling the behaviour of a hardware device in software terms, he needs to create a device class model describing the interface of the device, its run time behaviour and internal state. The model is materialised as an XML file that is validated against a static XML schema. The file is saved in the CCDB where additional processing takes place: the information from the XML document is shredded and distributed over a set of relational tables, and the existing devices (if any) of the device class in question are updated to match the new incarnation of the model (this process is known as *device promotion*). In the second phase the device classes are deployed on front-end computers. This is achieved by adding an element to an XML file validated against an XML Schema that is *generated dynamically* by an XSL transformation based on the directory of all available device classes provided by CCDB. The resulting XML file is also saved into the CCDB. In the third phase the device instances are created or updated. They are also modelled as XML documents compliant with yet another dynamically generated XML Schema. The XSL template used to generate this schema uses as its input the device class model created in step 1 augmented by other pieces of information coming from the CCDB, e.g. information on accelerator timing events.

The above demonstrates that FESA and CCDB are indeed very closely coupled and it is crucial to provide mechanisms ensuring the seamless integration of the two worlds.

## CONTEXT

The CCDB is not just a simple repository for FESA data; otherwise file system storage would have been sufficient. The information coming from FESA is only a small fraction of overall data managed by CCDB.

FESA devices are just one of several device types supported by CCDB (device names must be unique CERN-wide across all domains). CCDB makes FESA data available to external federated databases [3], augments it with additional information that is exploited by control room applications (device grouping, references and archives, meta-properties – properties that describe properties – for configuring GUIs), ensures the proper initialisation of FESA processes as part of front-end start-up sequences. In addition, CCDB provides much stronger system-wide coherency checks beyond local validations ensured by XML Schema in the FESA domain. Obviously this puts additional constraints on data that FESA produces (e.g. device name uniqueness).

## CHALLENGES

It is beyond the scope of this paper to discuss all possible solutions to the issue of XML/RDBMS impedance mismatch, but a few important problems are worth mentioning.

It is well known that XML and relational data models are not fundamentally compatible [4]. The XML is semi-structured, nested, hierarchical, polymorphic, and is in some cases based on varying XML Schema (the schema differs within the same class of documents that are validated against it). The relational data model on the other hand is highly structured, normalised, coherent and flat. Other XML features that map poorly to relational model include unbounded strings, variability (a parent element can have different kinds of children), and ordered polymorphic sequences (relational table rows are unordered unless explicitly sorted by a key and are not polymorphic).

In spite of that CCDB must provide means of saving and then restoring XML documents without altering their format. Internally it must parse and shred these documents and put atomic data elements into individual relational tables. These actions may trigger further data management events, such as device instance promotion. The database is also responsible for generating dynamic in-memory XML Schema documents out of information stored in the relational schema. FESA data must be updatable not only through dedicated tools, but also through general purpose database mechanisms. Between save and restore operations, data may be updated through CCDB specific mechanisms, outside of FESA realm.

## SOLUTIONS

In order to interface an XML application to a relational database one needs to provide an XML Processor acting as an interface between the two. This can be done in several ways:

- Embedded into the XML application as a library.
- Deployed as a middleware component on an application server.
- Coded as stored procedures inside RDBMS.

Up until FESA 2.10 the FESA/CCDB interface was provided in the form of a Java API (scenario 1), but for FESA 3.0 all functionality is going to be transferred inside database (scenario 3). Only a very thin Java adapter is left to integrate with the Java FESA tools.

Several representations are possible to store XML documents in a relational database. One is C*LOB representation* – the XML strings are stored as such in Character Large OBject columns. An alternative option is *composed representation* whereby an XML document is parsed and *shredded* in order to put individual values into database columns based on primitive data types. The *hybrid* approach is also possible where some fragments of an XML document are shredded while others are stored as CLOBs. CLOB representation enables easy storage and retrieval but is inefficient when it comes to querying or updating. Conversely, a composed representation requires a significant effort when shredding XML documents based on complex or variable XML schemas, but it is trivial to manipulate data through DML statements.

### *Initial Implementation*

In the early days of FESA project, the system was undergoing constant evolution. It went through more than 100 iterations in less than 2 years and most of them affected the specifications of the XML/CCDB contract. In the same period CCDB data model was also undergoing fundamental changes due to strategic choices for data management in accelerator controls.

Under these circumstances the only option was to use the CLOB representation augmented with very limited decomposition. Only the device class names, version numbers and front-end computer names were extracted and put into individual columns. Some rudimentary directory services were provided by the API to return them to the client applications.

Over the several subsequent releases of FESA an independent schema of FESA-specific relational tables was incrementally designed and created. It was fed with data resulting from partial XML shredding, loosely coupled with the core CCDB schema (no relational constraints, asynchronous synchronisation through database procedures); in parallel all XML documents were still stored as CLOB values for efficient retrieval by FESA tools without any loss of information. Due to the use of DOM (Document Object Model) parsing and XPath queries the performance of document saving left much to be desired. However, it was difficult to use the faster SAX (Simple API for XML) sequential parsing because the order in which information was stored in the XML files was not compatible with the required order of insertion to the database tables. On top of that the necessity of keeping the CLOB and composed representations in sync when performing updates from within CCDB was also tedious. A special XMLDB SQL syntax was used in order to update XML documents

inside database. This proved to be complicated and inefficient from the performance point of view. The problem worsened with the wide acceptance of FESA and the increasing number of operationally deployed devices. Each modification of a device class would trigger the update of all dependent devices, which could mean reading out, updating and writing back tens or hundreds of small XML files.

### *Current Evolution*

FESA 3.0 brings a big paradigm shift in the relationship between the framework and CCDB. The framework becomes decoupled from the CCDB until a developer decides to *deliver* his device classes into the operational environment. The responsibility for handling the deployment and instantiation of devices is taken up by the CCDB, while FESA class design is handled by an Eclipse IDE plug-in.

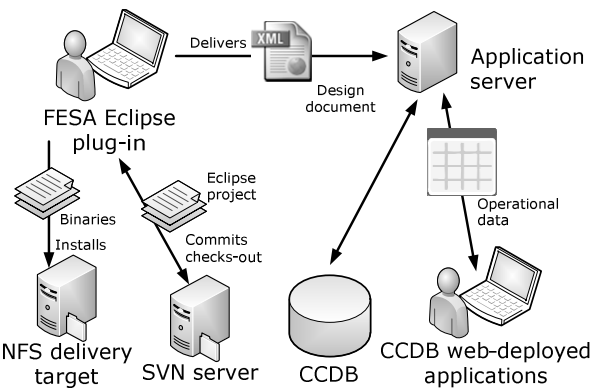


Figure 2: FESA 3.0 workflow.

The CCDB team took advantage of this opportunity to completely rethink the architecture of the FESA/CCDB XML processor. The CLOB representation was dropped in favour of the *composed* approach, for which the FESA part of the database schema was completely redone. The comprehensive suite of XML technologies integrated into Oracle 10*g* (XML DB) made it possible to move the XML processor inside database server.

For shredding the XML documents the choice was between two options:

- Register the XML Schema documents with the database, use the automatic shredding mechanism provided by XML DB, redistribute data from the resulting object relational tables into plain relational ones.
- Transform FESA XML files into Oracle *canonical* XML format using XSL transformations. The canonical format can be directly loaded into target tables using the supplied package DBMS_XMLSTORE.

The second option was preferred because the XML Schema documents for the deployment and instantiation data are generated dynamically and as such they are variable. Consequently, XSL stylesheets needed to be developed for all types of FESA XML. The choice for *recomposing* the FESA XML documents was between:

- Extracting canonical XML, piping it through XSL.
- Using SQL functions based on the SQL/XML standard, and some occasional XSL processing.

In terms of functionality and performance both solutions were deemed equivalent, and the development effort involved was similar. We decided to go for the second option because no additional XLS step was required in the case of recomposing the device class design XML documents. For the instantiation and deployment documents the limitation of SQL/XML specification had to be dealt with, as it does not allow for dynamic XML element names. To overcome this issue, intermediate XML documents are produced transformed with trivial XSL stylesheets to the final format.

## FUTURE WORK

The work for FESA 3.0 is still in progress and all specifications are not frozen at the time of writing. Although the major part of code for XML processor is already developed, it will inevitably go through several iterations until it will be released. However, we are confident about our choice of technology stack.

## CONCLUSION

Although FESA 2.10 is a production system that is widely adopted at CERN and beyond (GSI in Darmstadt), we can consider the road that has been covered until and including this version a learning process. When the development started the XML technologies were still considered to be emerging and the subject of XML/Relational mapping was not very well explored. On the RDBMS side, the support for XML was also evolving at a rapid pace.

The FESA data management was rethought and re-implemented. The technological choices for FESA 3.0 XML Processor, biased towards Oracle® solutions, are fully in line with the strategy in place for the CERN accelerator controls. Inevitably, these implementations are difficult to export to other environments.

## REFERENCES

[1] M. Arruat *et al*., "Front-End Software Architecture", ICALEPCS'07, Knoxville, USA, October 2007, WOPA04, p. 310 (2007)*.

[2] M. Arruat *et al*., "Use of XML Technologies for Data-Driven Accelerator Controls", ICALEPCS'05, Geneva, Switzerland, October 2005, PO2.094-5 (2005)*.

[3] R. Billen *et al*., "Accelerator Data Foundation: How It All Fits Together", ICALEPCS'09, Kobe, Japan, October 2009, TUB001.

[4] D. Draper, "Mapping between XML and Relational Data", article at http://www.informit.com/

* Published at the Joint Accelerator Conferences Website (JACoW) http://www.JACoW.org