# Case Study – XML Composer

## Use at a large USA State Government Agency

***By David Doyle, Independent Consultant and AllFusion Gen expert***

With the multitude of heterogeneous systems being used across the enterprise these days there is often a requirement to interface data between these systems. That is the challenge recently faced at a large client site. Two systems were being developed by different teams, one using .NET/SQL Server on the Win32 platform; the other using DB2/Advantage Gen on an IBM mainframe. [Aside: The mainframe system uses .NET and COM Proxies for its user interface but that is outside the scope of this article.] The majority of the data needed by the DB2 mainframe system originated on the SQL Server database and had to be added to the mainframe database in near real-time as it was updated in SQL Server for later processing by multiple mainframe application systems. Different analysis and policy goals resulted in the data on each platform being structured differently - both at entity and attribute level (different data formats, relationships etc) so a solution involving data replication would have been difficult to implement.

### XML and MQ

The chosen solution was to transmit each SQL Server transaction as a discrete XML document using MQ Series as the transport layer to guarantee delivery to the mainframe. Each XML document contains several mandatory attributes plus many optional attributes with multiple occurrences. Approximately 15 documents were defined - resulting in documents from 16K to 500K in size. The document size (often well over the 32K limit for Gen common format buffer) was another reason for the choice of MQ Series as the transport layer to route the data between the Win32 and mainframe platforms.

Using MQ as the transport layer means that the arrival of a document on the message queue can trigger a Gen server, an approach we used in one case, or a long running server could poll the queue for messages periodically. The latter is an approach we used for all the remaining messages as there is a large amount of reference data conversion/ translation between the two systems and it makes sense for the server to remain loaded throughout the day. We wrote a number of External Action Blocks in COBOL to access the MQ API for connecting to queues, getting messages, putting messages etc. After reading a document off the message queue we ended up with the document in a large group view (4092 chars * 250 occurrences). As an MQ message could not trigger a Gen transaction directly we wrote a very small COBOL program to execute the Gen transaction.

### XML Readers

XML Composer allows the user to define a 'reader' External Action Block into which is passed a group view containing an XML document. The exports of the EAB are regular Gen entity or work views along with a standard work view containing result information. As with Report Composer you generate the EAB 'stub' in Gen and then import it into CANAM's software. You then map the export views to a sample XML document or schema. A properties sheet for each attribute allows you to specify the data type, along with any formatting considerations or parsing requirements. As the timestamp data being sent was not in 'Gen' format we were able to customize the property sheet to map the incoming timestamps into Gen- format timestamps. After the mapping is complete a 'generate' button results in a dialog box where you specify the environment the code is being generated for COBOL, C or Java. In our case we generated COBOL code, which was FTPed to the host for compilation. The generated code is based upon the SAX parser commonly used to parse XML documents. At runtime success/fail parameters are returned along with the position of the failure in cases where the document is either not valid XML or is missing a mandatory attribute.

**XML Writers**

While our primary requirement was to produce XML readers, we also wrote one writer. The definition of the writer EAB is similar to the reader, the contents of incoming Gen entity or work views are written to a buffer formatted as an XML document according to a sample document or schema. You can create a schema within XML Composer itself while doing the attribute mapping but in most cases the document format will be dictated by the system providing or requiring the information.

**Version Control - Adoption**

What if the document changes after the initial definition? XML Composer has an 'adopt' option which allows creation of a new EAB based upon the definition of an existing one. When the ability to parse non-Gen timestamps became available as part of a patch to the product we used this feature to create a new version of our reader EAB using timestamp attributes instead of the text attributes that had been parsed in the original application code. The adopt facility matched 95% of the existing attributes allowing us to just map the changes.

**Positive Experiences**

Runtime performance was good - the largest (500K) documents taking around 2 seconds to parse on a heavily loaded mainframe. Technical support was excellent and responsive to the rare issues that arose.

## Footnote

*Since this case study was originally written, later versions of XML Composer now take advantage of the AllFusion® Gen plug-in capability to provide seamless, bi-directional integration between the two tools.*

*XML Composer is now capable of being registered as an AllFusion® Gen plug-in which allows the XML Composer toolset to be launched from within AllFusion® Gen. Once XML Composer has been launched in this fashion, a new XML Handler can be easily created based upon the AllFusion® Gen action diagram views.*

*The XML Composer Plug-in capability also allows the current AllFusion® Gen model to be interrogated in order to identify and select candidate external action blocks to be used as the starting point for a new XML Handler.*