Using XML with S88.02

David Emerson  
Sr. System Architect  
Yokogawa Corporation of America  
2155 Chenault Drive Suite 401  
Carrollton, TX  75006  
Tel: (972) 417-2753  
Fax: (972) 416-3966  
emerson@dallas.yca.com

KEY WORDS
IEC 61512, S88.01, S88.02, Batch, XML

ABSTRACT
The eXtensible Markup Language (XML) was released by the World Wide Web Consortium (W3C) in 1998 and has experienced rapid worldwide acceptance. When the SP88 committee started work on the S88.02 exchange tables XML technology was not available for consideration and relational database table technology was used as a basis for batch data exchange. As applications come to market based on S88.02 it is important to examine the newer XML technology for suitability for batch data exchange.

This paper provides a brief overview of XML and discusses how it can be used for batch data exchange. As an example an application that converts a control system's proprietary master recipes to and from XML is discussed. Lessons learned about the use of XML versus relational database technology are presented as well as possibilities for future uses of XML with batch control.

INTRODUCTION
When work started on the S88.02 standard in 1995 the SP88 committee evaluated various data formats and selected relational tables as the preferred format for exchanging batch related data. At the time the use of relational table technology was appropriate since relational databases were widely accepted, there were relatively low cost tools available to support them and they provided the complex data capabilities required by batch data structures.

Since 1995 a great deal has changed in the world of information technology, especially the emergence of Internet technologies. One of the most talked about new Internet technologies is the
eXtensible Markup Language (XML). While still in its infancy XML is experiencing rapid and wide acceptance, as evidenced by the steady introduction of new products that utilize XML.

As XML enters the mainstream of software tools it should be examined for use in the batch control industry. At the time this being paper was written, the S88.02 standard and its international equivalent IEC 61512-2 have been approved and are in the final bureaucratic motions of being released, however products based on the standards have not yet become widely released. So there is a window of opportunity to utilize a new technology like XML to implement batch data exchange building upon the work of S88.02, but without the complexity and difficulties of using relational technology for exchanging data.

**WHAT IS XML?**

The Extensible Markup Language (XML) was developed by the World Wide Web Consortium (W3C), the same group that maintains HTML. XML is defined by the W3C recommendation, REC-xml-19990210 Extensible Markup Language (XML) 1.0. XML is a data format; it does not specify the presentation of the data on web pages as HTML does. One of the keys to XML’s widespread acceptance has been its platform independence and its ability to support multiple languages and character sets using the ISO Unicode standard.

The XML syntax provides a straightforward method to identify data values with descriptions. For example in the following XML snippet it is easy to see that a Master Recipe contains a Header which in turn contains a Product ID which has a value of M101:

```xml
<MasterRecipe>
  <Header>
    <ProductID> M101 </ProductID>
  </Header>
</MasterRecipe>
```

This example could easily be conveyed in a text file, displayed on a web page, or e-mailed between collaborators and be human readable without the use of specialized software such as is needed to view data resident in relational databases. Although a similar file format could be, and in the past has been, constructed using text files, such as Comma Separated Value (CSV) files, the use of XML has advantage. Some of the advantages are that the file contains not only the data but also descriptions of the data and the data’s inherent hierarchy. This makes the use of XML much easier and robust.

Of course this is the good news. The bad news is that without a standard set of tags it is difficult for programs to process XML. One cause of this difficulty is that XML tags are case sensitive and must match exact spelling and that the tags are user defined. For example the following are all legal tags, but they are also all different tags:

```xml
<ProductID> <productID> <PRODUCTID > <Prod_ID>
```

So unless everyone uses the same spelling and capitalization it will be difficult to process each other's XML documents. However, the use of Document Type Definitions (DTDs) and schemas, help overcome this.

**DTDs & SCHEMAS**

The existing XML 1.0 recommendation and the April 2000 draft of the XML Schema recommendation define DTDs (Document Type Definitions) and schemas respectively. DTDs and schemas are tools that address the issue of precise tag spelling and capitalization, as well as many
issues not directly addressed in this paper. While the XML 1.0 recommendation defines XML DTDs and these are widely used, it has also been recognized that DTDs are lacking in some areas. Therefore the working draft for XML schemas has been developed and after some use and refinement is expected to become a W3C recommendation. For the purposes of this paper the term schemas has been used since it is assumed that they will gradually supplant the use of DTDs and enable more powerful XML applications.

A schema can be thought of as a definition of an XML based vocabulary. Using a schema, a set of tags and the hierarchy they may be used in can be defined. When tags are defined their precise spelling as well as their data types are set for the particular vocabulary. Likewise the hierarchy, grouping, order and other organizational features of XML documents in which the tags are used can also be specified in a schema. A schema can represent a customized language for an industry, company, or specific application. In fact there are currently efforts underway to create vocabularies for a number of industries and applications, including the financial industry, health and manufacturing industries.

Figure 1 shows how a schema can be used with an XML document. The schema and XML document are separate files, with the XML document referencing the schema. A software application that can read an XML document is referred to as an XML processor, this application could use the XML for any number of reasons including displaying it in a web browser or as input to a stand-alone program. The XML processor uses another piece of software called an XML parser, which reads the XML document, detects the reference to a schema, accesses the schema and verifies that the XML document complies with the schema's requirements. If the XML document complies, it is said to be valid. Once the parser validates the XML document the information in the document is passed to another part of the XML processor application where it is used.

The use of schema's permit XML to be extended by enabling the creation of other markup languages, each of which can be called a vocabulary.

While there are many subtleties and advanced features of XML, the concept of customized tags that describe data arranged in a hierarchy are some of the language's core strengths that make XML very well suited for use with batch data. The power of XML schemas to create specific vocabularies and enable the automatic parsing and validation of XML data adds value to the use of XML for the exchange of batch data.

The W3C continues to develop other XML related recommendations that add additional functionality to XML implementations. Of these the XSL Transformations (XSLT) recommendation and the Extensible Style Sheets (XSL) working draft facilitate the mapping and conversion, or
transformation, of one XML document into another form. Also expected to be helpful is the XML signature recommendation, which could improve security aspects, involved with transmitting ASCII data. These improvements and others that follow should increase the power of XML, further spread its adoption and provide additional tools and capabilities for exchanging batch data.

**S88.02 DATA EXCHANGE**

The upcoming S88.02 Batch Control, Part 2: Data Structures and Guidelines for Languages standard defines a method for exchanging batch control information between computer programs or systems. The method involves the use of relational tables or exchange tables as shown in Figure 2.

The S88.02 draft standard does not propose to define the internals of batch control, or other related systems. It states that only an interface specification is being defined, not the internal requirements for a system using the interface. Therefore the local data stores shown in Figure 2 may have different structures and contents in Tools A and B. The exchange tables represent a common format that can be used to exchange data.

The S88.02 exchange tables support four types of batch data:

1. Master and control recipe information
2. Process cell equipment information
3. Schedule information, and
4. Production information.

This list represents the most important and frequently handled data for batch control. However, no list can ever be 100% complete, especially over time as new needs develop. So the S88.02 draft also states that the exchange table definitions may be extended to support additional data. The additional data could be vendor specific data such as data addresses; end-user data that may support corporate business rules, or even some industry specific data such as may be useful for the pharmaceutical or other industry. The ability to extend the exchange tables is an acknowledgement that no one format can be all encompassing and fluid enough to handle all, or perhaps even most, applications over time. By permitting the expansion of the exchange tables the SP88 committee has enabled them to grow and be adapted as needed.

The selection of relational tables, as used in most common relational database management systems (RDBMS), was done in 1995 after careful examination of existing alternatives. Some of the
alternatives considered were text files, ISO 10303 (STEP/Express), and the Standard Generalized Markup Language (SGML). Using text files would require the creation of syntax and processing rules both of which would add to the effort but not add value to the end user. The STEP standard was not used due to the spin-up time, complexity and cost associated with it and the Express language. Likewise the SGML format would have required expensive tools and time to learn its complexity. In light of the alternatives, the use of relational tables permitted the minimum time required for learning new technology and relatively inexpensive tools (or at least readily available tools in most companies). However, at the time it was recognized that the use of relational tables:

- while expedient, involve a certain level of complexity,
- are actually better suited for storing data, rather than exchanging it,
- suffer from limitations of relational database management systems that despite an ANSI SQL standard actually differ in the syntax used and,
- requires the bridging of multiple operating systems using third party tools that can represent significant costs.

Despite the drawbacks, the use of relational tables was adopted since it represented the best and quickest path forward at the time.

The S88.02 standard also contains a high-level data model that specifies the objects and attributes and their basic relationships that cover the concepts of S88.01. While actually developed concurrently with the exchange tables the data model can be seen as the top level abstraction of batch control related data with the exchange table being one implementation based on it. During development of Part 2 the SP88 committee realized that there might be other implementations based on the data model in the future as new technologies emerged. One example of this is the OPC Batch Custom Interface Specification from the OPC Foundation, which defines an interface using Microsoft’s COM technology for the exchange of batch and equipment data. Another possible implementation could be done using XML, which would complement the existing methods as shown in Figure 3.

![Figure 3 - S88.02 Data Model Implementations](image)

In the future there will be multiple transport protocols available for batch data exchange, the capabilities of each will probably differ so the market will decide which ones become widely used and which fall into disuse.

**MASTER RECIPE DATA EXCHANGE**

Recent experience in the development of a software tool that converts proprietary master recipes to/from XML has provided insight into the applicability and feasibility of using XML for batch data
exchange. The conversion tool was implemented using Microsoft’s ActiveX and XML technology. An XML schema was developed to define the master recipe XML format and provide a basis for validating XML master recipes. The ActiveX conversion object is capable of reading in either the proprietary master recipe format or an XML format master recipe and converting it to the other format.

Internally the ActiveX conversion tool uses Microsoft’s Document Object Model (DOM) API that is part of Microsoft’s XML parser to build a copy of the recipe in memory. Use of the DOM permits easy construction of a memory resident tree structure the represents the internal structure of the recipe. Using the DOM API the recipe’s tree structure can be easily navigated.

The use of ActiveX technology resulted in a conversion object that can be accessed from various sources, such as web pages, C++ or Visual Basic programs. Depending upon the program, the conversion may be specified and triggered by a person or automatically based on a triggering event.

Since there is no existing standard XML schema for master recipes, the schema used by the conversion object is based on the S88.02 data model with some elements added as vendor specific extensions. While this schema actually has a proprietary format, it’s contents can easily be transformed to match other S88.02 compliant master recipe schemas using XSLT stylesheets.

A sample XML document that complies with the schema is shown below. This example only contains the recipe header in order to fit the size requirements of this paper, but the XML is still valid according to the schema, which demonstrates the power of exchanging recipe fragments using XML. If this were done in a relational database the actual data would not be human readable, would require a SQL query to retrieve and would require a larger size file, or set of files.

Sample XML Format Master Recipe Header

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MasterRecipe xmlns="x-schema:MasterRecipeV01.xdr">
  <Header>
    <ID> PROD100 </ID>
    <CreationDate> 2000-09-04T12:43:48 </CreationDate>
    <Version> 4.0 </Version>
    <VersionDate> 2000-10-10T16:23:03 </VersionDate>
    <VersionAuthor> Michelle </VersionAuthor>
    <ApprovalDate> 2000-10-12T13:04:22 </ApprovalDate>
    <ApprovedBy> Paul </ApprovedBy>
    <Status> Approved </Status>
    <StandardSize> 100.0 </StandardSize>
    <MaximumSize> 300.0 </MaximumSize>
    <MinimumSize> 50.0 </MinimumSize>
    <EngineeringUnit> KG </EngineeringUnit>
    <Comment> PRODUCT X </Comment>
    <Description> Production of Polymer X </Description>
    <ProductName> Polymer X </ProductName>
  </Header>
</MasterRecipe>
```

Some of the lessons learned from creating an XML master recipe schema and the conversion object were:

- Proprietary recipes will likely have custom properties. While custom properties can easily have a place in a standard format they will need to be handled individually when imported to a different system

- While the 88.02 exchange tables provide a standard set of tables and fields for master recipes, they should not be used as a literal basis for XML schemas. The table names and relationships
and fields in the S88.02 exchange tables were created to fit the capabilities and limitations of relational database technology. XML has a different set of capabilities and limitations so an XML schema should be designed to make the best use of XML, not SQL. Examples are the use of easily identifiable element names instead of cryptic and confusing table and field names, removing the use of foreign keys and tables required for 1:n relationships, and removing constraints placed on the SQL due to physical limitations of the number of columns in a table or characters in a field which are limits of specific database products.

- With XML it is much easier to identify and exchange fragments of recipes. For example it is very easy for a master recipe XML document to only contain a single recipe unit procedure or recipe operation. While this can be done in the S88.02 exchange tables it is not easy to identify, or find, the single fragment inside a database.

- Legacy Sequential Function Charts (SFCs) used to specify recipe procedural logic can be difficult to move to tables designed for S88.02 Procedure Function Charts (PFCs). While SFCs remain parts of recipes there will be a need for a significant number of custom, vendor specific, properties for each procedure. This will make the intelligent exchange and use of master recipes challenging.

**BATCH CONTROL MARKUP LANGUAGE**

The experience with XML to date has shown both the feasibility, and the need, for a set of XML schemas based on the S88.02 data model that would be used to exchange batch control related data. A logical set of schemas would cover:

- master and control recipes,
- batch schedules,
- equipment definitions, and
- production information.

This grouping would match the S88.02 data models. Collectively this group of schemas could be called a Batch Control Markup Language (BatchML or BCML for short) and used as the basis for exchanging batch control related data in a variety of environments and applications. It may also be wise to extend this work to cover the S95.01 standard for Enterprise – control integration.

Figure 4 shows how an industry standard schema could serve as the basis for many related schemas, each of which will differ, in hopefully minor aspects.
In this example a batch processing company may develop a corporate standard XML schema for master recipes. This corporate standard may use some of the company's in-house terminology, perhaps in the master recipe, schedule or production information corporate schemas. By doing this a batch processing company would give suppliers and vendors that do business with them the ability to view their data elements and hierarchies in order to provide a mapping with the suppliers or vendor's XML schema. Once a batch processing company's XML schemas are established then products from different vendor's could be interfaced with the corporate standard and enable the movement of batch processing data between systems and to external suppliers and customers as needed. Each vendor's XML schema may be customized for their product's special features.

While the use of many similar, but different XML schemas may appear confusing, the use of XSL transformation stylesheets can make conversion between similar schemas relatively straightforward. In addition commercial tools are available for configuring this mapping and automating transformations. This would surely make the integration of different applications within the plant floor and between the plant floor and business systems less costly.

**SUMMARY**

The S88.02 standard provides a solid basis for the development of different interfaces for the exchange of batch control data. The emergence and wide acceptance of XML in the information technology industry represents a valuable and well suited technology that can be used for batch data exchange.

As the OPC Foundation has done in the development of a Microsoft COM based batch data exchange interface, a new group could also use XML to create a Batch Control Markup Language. A Batch Control Markup Language coupled with new and existing tools and technology could be used to lower the cost of integration of batch control related, operational and business applications in the batch processing industry.
REFERENCES


*XML, A New Web Technology for Structured Data Exchange*, by David Emerson, presented at IMS Expo 1999