W3C WORKSHOP ON BINARY INTERCHANGE OF XML INFORMATION ITEM SETS

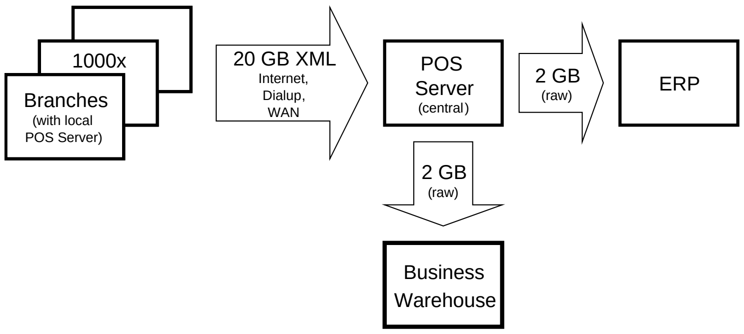
SAP Position Paper

INTRODUCTION

For 5 years, the XML 1.0 specification has had a great influence on modern document exchange. Designed as a subset of SGML, the design goals for XML 1.0 were focused on ease-of-use and human readability. But nowadays, XML documents are often used for data exchange without human interaction (on the XML 1.0 document itself). For such applications, the XML 1.0 serialization of the XML Infoset creates overhead on the wire and for processing, that might be avoidable by using a different and better suited serialization of the exchanged data.

REQUIREMENTS FROM REAL-WORLD BUSINESS SCENARIOS

To illustrate SAP's requirements for a more compact and faster processable serialization format of the XML Infoset we take the Point-Of-Sale Interface application (POS Interface), part of SAP's mySAP Retail solution. The goal of POS Interface is to make all items on all sale-slips of all stores of a given retail company available for central analysis and further processing.



For large retailers with some 1000 stores, there will be an average of 33 million sale-slips per day at an average of 3 items. In the current implementation, a local POS server aggregates all sales items of a store. The aggregated data is then sent in batch mode during closing hours to the central POS Server. The aggregation reduces the 100 million sales items to about 10-20 million, which results in about 2 GB raw data. The central POS server then forwards the collected data of all stores to the Business Warehouse and ERP system. In the current implementation, which is based on SAP standard interfaces and protocol (SAP IDoc), the upload and processing of the data will take about 2-3 hours. If the alternatively available standard protocol is used, which includes XML 1.0 data encoding, the data size is increased to about 20 GB.

Now, if the opening hours are extended or if the company opens stores in different time zones, the time slot for the batch jobs might shrink below the time that is needed for the actual processing.

A different problem occurs, if customers ask for better real time analyses. Sending more often smaller aggregations to the central POS server could meet this requirement. In worst case, each sales-slip will be sent directly to the central POS server. Hence, because no aggregation occurs in this case, the estimated size of the XML data is about 200GB per day.

Reducing the average message size and speeding up the processing is extremely helpful in this scenario, because this will allow better real time analyses without increasing the overall system landscape, for example, by adding additional application servers.

RETAINING XML INFOSET OR ADOPTING FULL BINARY ENCODING

As shown above, decreasing the average message size and speeding up processing allows us to build applications that are not feasible with respect to the cost of current system landscapes. Several approaches have been discussed internally:

Compression

Compression of the XML 1.0 documents (gzip or other) only reduces the message size on the wire but increases the processing time.

Binary encoded XML Infosets

This approach retains the XML Infoset but chooses a binary and hence more compact and faster processable serialization format. One advantage of this approach is that only the XML parser must be rewritten but all tools on top of it can remain unchanged. This will also enable the processing of XML instances without in-advance XML Schema knowledge (XPath, XSLT,…).

Full binary encoding

For a full binary approach, both peers must agree in advance on the binary format of the exchanged data. This technique is well known from RPC, DCOM, CORBA and RMI and seems to be the fastest approach. However, since this binary data is not self-describing, generic XML techniques can not be applied directly. Additionally, to avoid data mismatch a versioning mechanism must be added, that is, both the sending and the receiving application always have to know the XML Schema that is to be used..

LOOSELY OR TIGHTLY COUPLED

The decision on which of these approaches to use highly depends on the requirements of the given business scenario.

In a tightly coupled system where a client directly talks to the server and the overall system is managed centrally, the full binary encoding seems to be the best suited approach. All extra information to keep the XML Infoset is unnecessary and will result in slower performance.

If the system is loosely coupled or if generic intermediaries should perform extra processing steps on the messages, the binary encoded Infoset approach might be better suited.

CONCLUSION

The POS Interface does not fit well into the two categories above. On the one side, POS is a decentral system, and XML was chosen for easy integration of point-of-sale terminals of different vendors. On the other side, POS Integration is often carried out for thousands of stores. As a consequence, the interfaces do not change that often and the exchanged data should be as compact as possible.

A solution should

- Meet the requirements from real-world scenarios
- Fit into the existing stack of Web services and XML standards such as SOAP and WSDL
- Be easy to implement

ANSWERS TO THE W3C QUESTIONAIRE

Q: *What work has your organization done in this area? (We are particularly interested in measurements!)*
A: For many years, SAP has established several internal communication protocols such as Intermediate Documents (IDoc) and Remote Function Calls (RFC). Additionally, we have made SAP business logic available via Web services. Our measurements indicated that SOAP-based XML messaging is 2-3 times slower compared to our native communication protocols.

Q: *What goals do you believe are most important in this area? (e.g. reducing bandwidth usage; reducing parse time; simplifying APIs or data structures, or other goals)*
A: Our goal is to optimize the overall processing time, including both transmission and parsing time while allowing generic tools to manipulate and transform the exchanged business documents.

Q: *What sort of documents have you studied the most? (e.g. gigabyte-long relational database table dumps; 20-MByte telephone exchange repair manuals; 2 KByte web service requests)*
A: See scenario description. The range is between some KBytes - in case the point-of-sale terminal sends each sales slip directly to the central POS Server - to several GBytes - in case the sales slips are accumulated by the local POS Server and sent in bigger chunks to the central POS Server.

Q: *What sorts of applications did you have in mind?*
A: All types of business applications provided by SAP e.g. CRM, HR, SCM, Financials, etc. See the scenario description for a detailed example.

Q: *If you implemented something, how did you ensure that internationalization and accessibility were not compromised?*
A: Considered only at business application level.

Q: *How does your proposal differ from using gzip on raw XML?*

A: Since the technology that meets our requirements depends on the actual business scenario, we have consciously not specified appropriate solutions in more detail. Thus, a simple comparison to a simple compression technology is not possible. Our experience with SAP-internal communication protocols showed us that compression helps on low-bandwidth connections, but is counterproductive in high-speed local networks. Therefore, the wire format of business documents must meet the following requirements:

- Must be compact (reducing bandwidth and memory)
- Must be faster to parse (in comparison to XML 1.0)
- Must allow generic processing (XSLT, XPATH)
- Must be derivable from XML Schema
- Must be easy to implement (by devices with reduced processing power)

Hence, we believe that neither simple compression nor binary protocols (e.g. protocols like RPC, DCOM or IIOP) will fit our requirements for all business scenarios even though they may be applicable for some selected scenarios.

Q: *Does your solution work with any XML? How is it affected by choice of Schema language? (e.g. W3C XML Schema, DTD, Relax NG)*

A: We consistently make use of XML and XML Schema. Other schema languages have not been used.

Q: *How important to you are random access within a document, dynamic update and streaming, and how do you see a binary format as impacting these issues?*

A: In the area of business application integration, random access and update of XML documents is mostly needed in intermediary applications, only, for example, for mapping and transformation between different XML business document standards. Thus, non XML-based encodings can only be processed with explicit knowledge of the corresponding XML schema which might not be available for all intermediaries.