B2B Protocol Standards and their Role in Semantic B2B Integration Engines

Christoph Bussler
Oracle Corporation
Redwood Shores, CA 94065, USA
Chris.Bussler@Oracle.com

1 Introduction

Corporations need to exchange business data to conduct business with their trading partners. Large corporations with sometimes millions of data exchanges every day realized a long time ago that an electronic transfer of business data between their internal software systems and those of their trading partners has many advantages over manual transmission through fax or phone. Some of them are reliability, timeliness, security, scaleability and traceability.

The idea of electronic transmission triggered the development of business-to-business (B2B) protocol standards for business data exchange like EDI [1,2,3] and SWIFT [4] over value-added networks (VANs) [6,9,11,17] over 25 years ago [5]. For example, EDI defines the syntax and the semantics of messages exchanged. "Syntax" refers to the message layout and "semantics" to the valid data types and consistent vocabulary (data type values) used in messages. Each message is either a message with the intent of action (like "new purchase order") or an acknowledgment message indicating the successful transmission of a message ("received purchase order") or an error message indicating an error situation ("reject purchase order"). The interpretation of the intent of action has to be standardized, too, in order to guarantee correct behavior from trading partners. For example, if a new purchase order is accepted then the sending trading partner expects the delivery of the ordered products in the specified time frame. In addition, each message has at least two parts: a header containing meta data about the message itself like the sender and recipient as well as the payload containing the business relevant information.

B2B protocol standards like EDI and SWIFT are well-defined and well-established in the industry in the sense that they not only provide a defined syntax but also a defined vocabulary (in conjunction with data types) for values of the message fields. Infrastructure is in place today (software as well as networks) to deploy and to use those standards. Standard organizations are keeping the standards up-to-date with changing and expanding business requirements [19,20]. Once implemented by trading partners, these standards allow the reliable exchange of business messages. As soon as a company has implemented those standards it can participate in the global message exchange with little effort. The implementation itself, however, was costly since VANs charge a fee to use the networks and the software implementing the standards was expensive.

In recent years two major "phenomena" changed the situation promising almost "free" business data exchange: the adoption of the Internet [12,13] as communication medium and the development and wide-spread use of XML [11]. The Internet promises cheap ("free") and widely available communication while XML

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

promises the easy definition and implementation of documents and messages exchanged over the Internet. The basic idea was to replace old technology with new one: using the Internet for communication rather than VANs and the use of XML as a tagged syntax of describing messages instead of existing formats like positional or comma delimited syntax. The belief is that this combination of the Internet and XML makes it a lot “easier” (i.e. cheaper, faster) to engage in business-to-business communication then the way defined by standards like EDI or SWIFT.

However, as can be observed by looking at the work of B2B protocol standard organizations [14,16], the “easier” is not that easy after all. Using the Internet might be cheaper from an accounting viewpoint, but it does not provide all the functionality required for reliable message exchange. Security, availability, privacy and service levels are some of the properties required for serious message exchanges [8] that go beyond what the Internet or XML provide. XML is - after all - a notation for syntax. It was not developed to define semantics (like vocabulary), consistency (valid use of vocabulary), security, the description of message exchange sequences or the definition of correct interpretation of exchanged messages.

As a consequence, research and development in network security is increasing as well as the deployment of virtual private networks (VPNs) [21,22]. Standard committees developing B2B protocols spending most of their time developing a framework around XML to address the functionality necessary not originally provided by XML [14,16]. Especially existing standard committees and corporations supporting B2B protocol standards like EDI and SWIFT are working on defining the XML representation of their current semantics making the point that XML addresses syntax only (see swiftml [7] and EDI [18]).

Architectural components that execute message exchanges according to B2B protocol standards at runtime are called B2B protocol engines. A B2B protocol engine is a necessary but not sufficient component in trading partner communication over networks. Once messages are received they need to be processed and before messages are sent they need to be generated. The activity of processing or generating messages involves the back-end application systems storing the business data. A B2B integration engine is the software component that connects a B2B protocol engine with one or several back-end application systems to provide an end-to-end solution.

This article will first focus on B2B protocol standards and B2B protocol engines. Afterwards the embedding of B2B protocol engines in B2B integration engines will be briefly discussed.

2 B2B protocol standards and B2B protocol engines

A B2B protocol standard in general is the description of the message formats exchanged (e.g. purchase order), bindings to transport protocols (e.g. HTTP/S [23]), the sequencing (e.g. after sending a purchase order message a message acknowledgment must be received), the process (e.g. after a purchase order was send a purchase order acknowledgment must be received), the security to be provided (like encryption, non-repudiation) and many more properties. Some standards focus only on a subset of the properties, others provide recommendation for the complete set.

If not every aspect of a standard is defined by the B2B standard definition itself then each trading partner has to agree individually with each of it’s trading partners on an implementation of the aspect. For example, if a standard only defines document types, but no transport binding, then two trading partners need to agree on the transport binding in order to make their systems interoperable.

B2B protocol standards isolate the back-end systems from the messages sent or received over a network. This requires that a B2B protocol standard defines the content of the messages in terms of syntax and semantics in order to be independent of the back-end application system’s syntax and semantics of data. For example, in case of purchase orders, a standard needs to define the elements of a purchase order like company identifier, address (bill-to, ship-to) and line items. Part of this is the definition of valid values for the elements. This not only comprises individual element values but also any consistency constraints between different values. Also, it needs to define the syntax used to express content like XML, comma-delimited or positional (column - row addressing).

The term “message” is used in context of a real exchange of content over a network. A message contains header information as well as payload, i.e. everything required to direct the message to the correct recipient and everything required for the recipient to interpret the message correctly if no error occurred. The term “document” is used for the pure business content. For example, the description of the elements of a purchase order as such is a document type. It is not concerned about header information or any other data than business data content. Usually the payload of a message is of a particular document type. “Business event” is used to express the intent. For example, “create” purchase order in comparison to “update” or to “delete” purchase order. All these three business events have a different effect in the enterprise.

The same type of business data might be defined by several B2B standards (e.g. purchase order in OAGIS as well as RosettaNet) even within the same industry. A trading partner has to deal with all standards it’s trading partners require and that might mean that one trading partner has to deal with several standards at the same time. In case one trading partner exchanges messages across industries, the variety of standards is likely to increase even more. The multitude of purchase order standards is not really a benefit since one trading partner might not be able to standardize on any of those but has to entertain all of them at the same time.

The following section will provide a set of criteria that can be used to classify B2B protocol standards. As can be seen there are many components to a B2B standard far exceeding the abilities of XML or any other syntax. The examples given do not provide a full coverage of all B2B standards available or currently under development by standard committees. They are purely chosen to illustrate the classification.

2.1 Classification of standards

Before going into the detailed classification of B2B standards as such a higher level classification separating B2B standards themselves has to take place. The B2B standards and examples discussed so far described messages that had a certain intent. For example, the fact that a trading partner receives a purchase order indicates that a buyer wants to buy some goods and expects a purchase order acknowledgment as well as an invoice and the actual delivery of the goods. These type of B2B standards are termed “business event B2B standards” since the messages are interpreted as events that cause action in back-end systems.

However, this is not the only type of interactions between trading partners. Sometimes a trading partner would like to retrieve information from several back-end system as sources of information. In this case information syndication takes place. Standards that support the syndication of data are termed “syndication B2B standards”. A good example are news distributing organizations [24,25].

A third type of standards are “supporting standards”. These standards support the development of B2B standards although they can be used in other developments, too. For example, XML is of this nature. The XML standard can be used as the syntax for B2B messages, but also as a standard for defining the syntax of data stored in a database.

2.2 B2B protocol standards

The following discussion provides a list of classification criteria for B2B protocol standards. The criteria chosen do not include network protocol properties like those discussed in [26,27] since B2B standards use network protocols as transport mechanism rather than being alternatives. Not all B2B protocol standards implement all criteria. For example, OAGIS [28] does not provide a process definition between trading partners, but RosettaNet [29] does by means of Partner Interface Processes (PIPs).

Document types

Document types define the payload supported. XML Schema [30] and DTD [31] are popular ways of defining document types. Through both mechanisms the structure of a document is defined in terms of fields and substructures. For example, a purchase order contains a ship-to and bill-to address as well as line item definitions. OAGIS, RosettaNet and EDI are examples of standards defining the payload.

Semantics

Semantics has many facets in context of B2B protocols. These are

Business content (vocabulary): An example of business content is a product name. For example, a product name has to match the receiver's valid product names so that an order can be processed correctly by the receiver. In addition, not every possible value is a correct one and consistent with other values in other fields. Consistency rules might have to be defined between the fields of a document (and enforced during execution). Several approaches exist to define business content.

- Defined by standard itself. In this case the valid field values are defined by the standard itself. For example, RosettaNet provides a business dictionary, EC technical dictionary, IT technical dictionary [29] that contains valid values. Sender as well as receiver have to map their internal values to those as defined by the standard.
- Defined by receiving trading partner. In this case the receiver requires its vocabulary to be used and the sender has to make sure that it uses the receiver's vocabulary correctly.
- Defined by sending trading partner. In this case the sender uses its vocabulary and leaves the translation into the receiver's vocabulary to the receiver.
- Trading partner identification. Trading partner identifications are part of messages sent. They also need to be understood by the participating trading partners and so they have to be standardized analogously to business content. Some standards define a particular naming schema to be used, for example, DUNS numbers [32].
- Message identification. Message identifiers need to be agreed upon like trading partner identification. Both, sender as well as receiver need to interpret the identifier the same way.

Constants: Units of measure [33] and country codes [34] are also values that need to be agreed upon by either trading partners or defined by a standard itself.

Data types: Business contents can be expressed in data types. For example, an address can be defined as a structure of name, street, city, zip and country. Standards using XML schema [30] as the document definition language can define data types.

Intent: A trading partner receiving a message needs to interpret that message in context of its back-end systems. The message has to be stored in the back-end system and the back-end system has to interpret the message according to its intent. Interpretation consists of the

- Correct interpretation of vocabulary. The business contents of the message must be interpreted according to the agreement with the trading partner. This requires the correct translation into the back-end system's vocabulary.
- Correct interpretation of message and correct initiation of action. A trading partner receiving a consistent and correct message is expected to act according to the message content. For example, if a purchase order is received then a purchase order acceptance document has to be sent as well as the goods delivered.

Transport binding

The transport binding defines how a message to be sent to a trading partner is encoded within the rules of a network protocol like HTTP/S [23], S/MIME [35], FTP [36], EDIINT [37] or beep[26]. Not all protocols define a transport binding. For example, OAGIS does not define a transport binding. Others define only a binding without defining any document types. SOAP [38] falls into this category. RosettaNet defines a transport binding as well as document types.

Message definition

In addition to encoding of the document itself headers have to be defined as well as marshaling rules to package the whole message. Headers and marshaling define how the complete message layout looks like when sent over the network protocol.

Exchange sequence definition

- Asynchronous messages sent over a network protocol are in most cases acknowledged (either positively or negatively in case of error). The exchange sequence definition says when acknowledgments are sent and the time limits for it. It also defines any retry logic, i.e. when to retry and how often.
- "Synchronous messages" (i.e. synchronous invocations) over a network protocol have to be acknowledged, too. However, in this case the acknowledgment is sent as a return parameter and not (as in the previous case) as a separate asynchronous message.

Process definition

On a business level business event behavior is defined. For example, a purchase order has to be followed by a purchase order acknowledgment. This exchange on a business level breaks down into several messages as defined by exchange sequences. This business level process definition is between trading partners only. It does not define the processing within a trading partner.

Security

A receiver of a message needs to know that the message is coming from the receiver that claims to have sent it, that the message was not delayed or repeated, that the message was not altered and that the message was not generated from an entity other than the sender. Furthermore, for a receiver it is important to be able to proof at any time that the message was sent by the receiver. This way the sender cannot deny having sent the message. Several security mechanisms are necessary to enable this behavior [39].

Syntax

Since the appearance of XML it is in many cases, if not in all of them, chosen as syntax for data exchange between companies. However, XML is not the only syntax available.

- Tagged document format. XML falls into the category of tagged syntax since the values are enclosed in matching tags. Non-xml tagged syntax exists, too.
- Positional document format. In this case the values are within certain positions in a document. For example, between the 4. and 10. column on line 2.
- Delimited document format. In this case the values are separated by delimiters. Delimiters can be the same (like ",") or can be keywords.
- Binary document format. In this case the values are encoded and without program support the document contents cannot be read easily by a human.

Trading partner specific configuration

In many cases companies use B2B protocol standards as defined by the standards organization. However, in some cases trading partner specific modifications need to be applied to support trading partner specific requirements.

- B2B protocol modification. In this case the B2B protocol will be modified. For example, specific fields are added to a document type or specific fields will be deleted. EDI uses guidelines for defining the modification. A guideline defines which parts of a document type are used for transmission. GXML [40] is a language for defining guidelines.
- B2B protocol configuration. Some standard organizations realized that trading partner specific configuration is required. For example, time-out or retry intervals are sometimes trading partner specific. RosettaNet, for example, allows the trading partner specific configuration like time-out, retry interval and retry counts.

Exactly-once behavior (or at least once or best effort) is not mentioned explicitly since the sequencing together with the process definition and the message meta-data can implement this.

Each B2B standard either has to determine its settings for the above discussed criteria or the trading partner have to agree on those individually. If neither is done, interoperability between two companies becomes very difficult since some of the settings are decided arbitrarily at deployment or implementation time.

2.3 Samples of B2B standards

There are many vertical industries and each industry has in general several B2B standards in use or in development. It is almost impossible to provide a complete list of all standards available. At [41] a comprehensive coverage of standards can be found from almost any vertical application domain. Table 1 lists only a few of all available standards and those under development. It illustrates the existing variety. A “B” under “Type” indicates a business event standard, a “S” a syndication standard and a “U” a supporting standard.

Table 1: Examples of Business Event, Syndication and Supporting B2B Standards

Standard	Long Name	Type	Reference
BizTalk		B	[44]
Beep	Blocks Extensible Exchange Protocol Framework	U	[26]
cXML	Commerce XML	B	[48]
ebXML	Electronic Business XML	U	[46]
EDI	Electronic Data Interchange	B	[1,2]
fPML	Financials Products Markup Language	B	[49]
GISB	Gas Industry Standards Board	B	[50]
gXML	Guideline XML	U	[40]
ICE	Information and Content Exchange	S	[24]
OAGIS	Open Applications Group Integration Specification	B	[28]
OBI	Open Buying on the Internet	B	[47]
OTA	Open Travel Alliance	B	[43]
RosettaNet		B	[29]
RSS	RDF Site Summary	S	[25]
S2ML	Security Services Markup Language	U	[51]
SOAP	Simple Object Access Protocol	U	[38]
SWIFT		B	[4]
tpaml	Trading Partner Agreement Markup Language	U	[52]
UDDI	Universal Description, Discovery, and Integration	U	[53]
WSDL	Web Services Description Language	U	[54]
xCBL	Common Business Library	B	[45]
xkml	XML Key Management Specification	U	[55]
XML	Extensible Markup Language	U	[31]
XML Schema		U	[30]
XML-RPC	XML Remote Procedure Call	U	[15]
XP	XML Protocol Activity	U	[56]

2.4 B2B protocol engines

B2B protocol engines are software components that execute actual message exchanges over networks according to B2B standard specifications. Sophisticated engines have to support any existing B2B protocol standard. Since some standards are under development and not finalized yet a B2B protocol engine should be able to dynamically add not yet defined standards over time. In addition, sophisticated B2B engines allow the customization of B2B protocol standards on a per trading partner basis. Some trading partners do not use B2B protocol standards themselves but have a proprietary protocol defined. A B2B protocol engine must be able to implement proprietary B2B protocols as well. In addition to the B2B protocol standards they might provide additional functionality to required by B2B standard definitions like the persistent storage of received messages (reliability), duplicate detection (dependability) or non-repudiation (security).

B2B protocol engines need to know trading partner specific configuration information. For example, each trading partner has to define a network address (e.g. URL) where message are received. Trading partner information like network addresses as well as identifying information like DUNS numbers are stored in a trading partner profile management component.

3 B2B protocol engines and B2B integration engines

A B2B protocol engine masters the communication between trading partners only. It neither interprets the messages received nor extracts or inserts business data into back-end application systems. Interpretation of messages and communication with back-end systems is done by B2B integration engines. B2B protocol engines are components of B2B integration engines. The example given next will help understand the overall data flow and the B2B integration engine architecture shown subsequently.

In general, two overall directions of data-flow can be distinguished: "outbound" and "inbound" from a trading partner's internal viewpoint. Outbound refers to the case where business data originate from a back-end system and have to be send out to one or more trading partners over a network. Inbound refers to the case where a message is received from a trading partner and needs to be interpreted and inserted into a back-end system (or rejected if invalid). One trading partner's outbound message is another trading partner's inbound message.

One of the "famous" examples is the business data pair purchase order (PO) - purchase order acknowledgment (POA). A PO and a POA are related by the POA acknowledging that the PO was received, is correct and is accepted (i.e. products will be delivered). For the trading partner that initiates the PO it is important to receive the POA. The trading partner receiving the PO needs to generate the POA to send it back. This pair, PO - POA is a "round-trip" exchange from the initiating partner: one message sent (PO) and one corresponding message received (POA). Figure 1 puts the example into context of the B2B integration engine's functionality. The figure points out that the formats extracted from and inserted into the applications are different from those sent over the network. PO' and POA' are in the format of the particular B2B protocol deployed between the two trading partners.

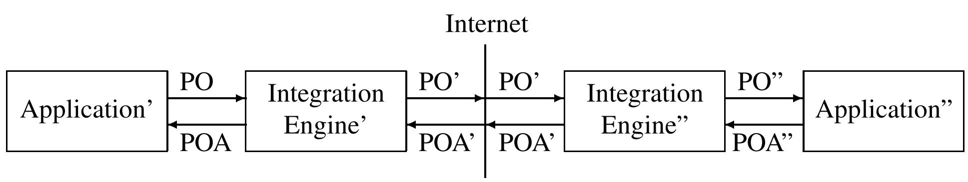

Figure 1: Purchase Order B2B Exchange ("round-trip")

A B2B protocol engine is responsible for sending out a message to a trading partner and receiving back an acknowledgment message (like the PO - POA pair). Application adapter technology is used [42] to extract business data from back-end applications or insert business data into back-end applications. Workflow technology is deployed to take the business events coming from adapters and initiate an appropriate workflow (e.g. approval workflow). The successful execution of a workflow causes the business event given to the B2B protocol engine that in turns transforms the business event into a message and sends the message to the trading partner. Instead of workflow system queueing can be used if no business logic is necessary. Figure 2 represents the basic components of an B2B integration engine and its communication with applications.

Figure 2 shows that the back-end systems of the trading partners involved are not aware of each other. It also shows that two trading partners can communicate with each other as long as they are compliant to the same B2B protocols. It is not necessary that they have the same implementation of a B2B integration engine installed (indicated by the dashed box). The B2B integration engine with the help of the B2B protocol engine isolates the back-end system from the external world by means of the B2B protocols. More specifically, each trading partner involved has to send the messages in the format as described by the B2B standard independently of the particular back-end system used. This isolation of the back-end systems by means of the B2B protocols characterize a B2B environment. If the back-end systems had to produce the format as understood by the trading partners back-end system, then an IAI (Internet application integration) solution would have been built. The implication is clear: each trading partner would have to implement as many formats as there are back-end systems at its trading partner's sites.

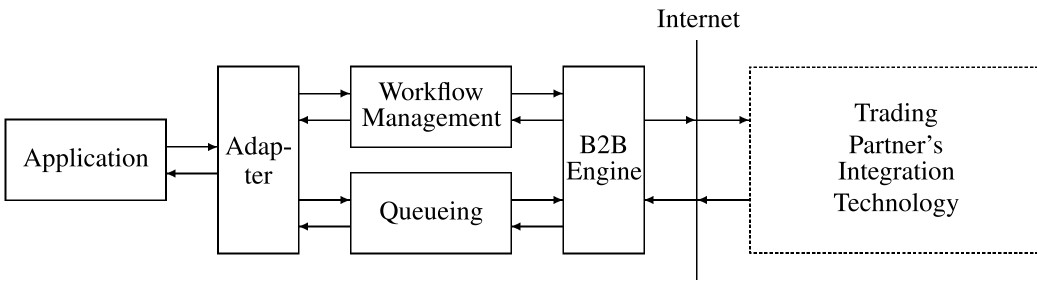

Figure 2: Components of a B2B Integration Engine

B2B protocol standards however only address part of the problem. They isolate trading partners from each other with a controlled syntax and semantics. However, one trading partner might have to trade with other trading partners through many B2B protocols. In this situation a trading partner has to deal with many B2B standard protocols at the same time. If the same business document (like a PO) is traded using different B2B standards then the trading partner has to translate every PO from every B2B standard into each of its back-end systems.

4 Summary and outlook

Looking ahead the number of B2B protocol standards can be expected to be in the low 100's. This rather high number is a result of vertical application domains creating not only their own standards but several standards within one application domain. At this point the question arises what "standard" means in the area of B2B. Currently, many standards are proposed by standards organizations, consortiums, individual or a small group of companies. These standards sometimes overlap or even compete with each other (for example, RosettaNet and OAGIS in the definition of document types). Consequently "standard" does not mean "globally unique definition" any more.

Without further commenting on the development and the appearance of standards as such, this development has an important implication for B2B integration engine technology. It is very desirable (almost a "must") that B2B protocol engines as well as B2B integration engines are agnostic to B2B protocol standards. If implemented with this in mind, any new B2B protocol standard can be dynamically added and supported by a B2B protocol engine. Following this approach a company deploying a B2B integration engine can add standards over time as required by trading partners being added.

5 References

[1] www.x12.org
[2] www.unece.org/cefact/index.htm
[3] www.unece.org/trade/untdid/welcome.htm
[4] www.swift.com

[5] www.swift.com/index.cfm?item_id=1243
[6] www.swift.com/index.cfm?item_id=3184
[7] www.swift.com/index.cfm?item_id=2642
[8] www.swift.com/index.cfm?item_id=2160
[9] www.gegxs.com/geiscom/template.jsp?page=1&id=44
[10] www.ipservices.att.com/products/product.cfm?productid=edi&disp=techspecs
[11] www.w3.org/XML/
[12] www.fnc.gov/Internet_res.html
[13] webopedia.internet.com/TERM/I/Internet.html
[14] lists.w3.org/Archives/Public/xml-dist-app/
[15] www.xmlrpc.com/
[16] discuss.develop.com/soap.html
[17] www.edi-info-center.com/html/vans.html
[18] www.edi-info-center.com/index.html#XML
[19] www.x12.org/x12org/Subcommittees/dev/dmwkreqform.doc
[20] www.x12.org/x12org/Subcommittees/dev/Y2K.htm
[21] www.internetwk.com/VPN/default.html
[22] kubarb.phsx.ukans.edu/~tbird/vpn.html
[23] webopedia.internet.com/TERM/S/SSL.html
[24] www.icestandard.org/
[25] www.egroups.com/files/rss-dev/specification.html
[26] www.ietf.org/ids.by.wg/beep.html
[27] www.w3.org/Protocols/
[28] www.openapplications.org/
[29] www.rosettanet.org
[30] www.w3.org/XML/Schema
[31] www.w3.org/XML/
[32] www.dnb.com/dunsno/dunsno.htm
[33] www.unece.org/cefact/rec/rec20en.htm#Top
[34] www.unece.org/cefact/locode/loc002.zip
[35] www.ietf.org/html.charters/smime-charter.html
[36] www.w3.org/Protocols/rfc959/
[37] www.ietf.org/html.charters/ediint-charter.html
[38] www.w3.org/TR/2000/NOTE-SOAP-20000508/
[39] Stallings, William: Network Security Essentials: Applications and Standards, Prentice Hall, ISBN 0130160938
[40] www.edifecs.com/b2b_resources_guideline.jsp
[41] www.oasis-open.org/cover/
[42] java.sun.com/j2ee/connector/
[43] www.opentravel.org/
[44] www.biztalk.org
[45] www.xcbl.org/
[46] www.ebxml.org/
[47] www.openbuy.org/
[48] www.cxml.org/home/
[49] www.fpml.org/
[50] gisb.org/imp.htm
[51] www.s2ml.org/
[52] www-106.ibm.com/developerworks/library/tpaml.html
[53] www.uddi.org
[54] msdn.microsoft.com/xml/general/wsdl.asp
[55] www.verisign.com/developer/xml/xkms.html
[56] www.w3.org/2000/xp/