Establishing an Inter-RTO Web Service Messaging Standard

14 October 2005

Contributors:

Carl Ozkaynak

Maxime Pitard

Dick Brooks

Table of Contents

• Overview

Seamless interfacing with business partners is the key component of a successful operation. This is especially true in the case of Regional Transmission Operators (RTOs) who must interface with a variety of entities including other RTOs.

The use of Web Services for Business to Business (B2B) inter-RTO communication is a critical component for RTOs and RTOs are looking to interoperate via Web Services because it provides a true standard interfacing solution to integration. Previously this task was very difficult because in order for systems to inter-operate they needed to have a deep understanding of how each parties system worked and each had to understand custom APIs and this was generally more trouble then it was worth.

Web Services provides an approach to having software work together based on agreed upon open standards where communication is based on a common language, an agreed interface (do not need to worry about how the functionality is implemented), and easy for those who need the service (to find and employ it).

In moving to Web Services an organization will need to establish or adopt standards for interoperability and for exchanging business messages.

This paper gives an overview of how the WS-I Basic Profile, ebXML Messaging Service (ebMS), and Message Exchange Patterns (MEPs) simplifies this task as well as provides a framework for establishing the foundation for a successful Web Services data exchange.

• Introduction to RTOs

On December 15th, 1999 the Federal Energy Regulatory Commission (FERC) issued Order 2000 and it marked the beginning of a move toward a national transmission system that would be independently operated. Utilities that own, operate or control transmission lines (the lines that carry power from generating plants to substations) were encouraged to turn control of the lines over to an independent agency (Regional Transmission Organization). The rationale for this is that, as the industry undergoes deregulation, an independent entity could ensure equal treatment to all industry participants using the transmission lines and increase the reliability of the system by placing a few entities in control of the national wholesale bulk electric system rather than having entities with varying vested interests control the lines. In order for an organization to become an RTO, the organization must receive FERC approval before it can be designated an RTO. To receive this approval, it must meet the characteristics and functions outlined in Order 2000 (http://www.ferc.gov/industries/electric/indus-act/rto.asp).

• Introduction to Wholesale Electricity Markets

Wholesale Electricity market operations involve several complimentary functions intended to ensure the reliable delivery of electricity, wherever it's needed, at the most cost effective price. Auction based market mechanisms have been implemented by some independent clearinghouse companies that serve as an "exchange" for electricity buyers and sellers to

consummate financial transactions worth billions of dollars annually. These "exchange functions" are frequently performed by "Independent System Operators" (ISO) or RTO's. Just as auditability, integrity and reliability are critical success factors for financial institutions (e.g. banks, brokerages, stock exchanges, etc.) this is also the case for ISO/RTO companies that operate wholesale electricity market exchanges.

The mechanisms used by ISO/RTO companies to ensure auditability of financial transactions vary from market to market. Some essential processes and procedures, intended to ensure financial auditability, are documented in the Sarbanes-Oxley Act of 2002 [SOX] and Public Accounting and Oversight Board (PCAOB) auditing standards [PCAOB-STD].

From its inception the ebXML initiative [ebXML] set out to create reliable and auditable end-to-end processes to enable companies to consummate legally binding business transactions over the Internet. In designing the ebXML standards the following assumptions were considered:

Each legal entity will have some form of software that will enable them to engage in business-to-business communications. The challenges' facing these parties include:

- Data is exchanged between companies using wide area, and possibly public, networks (e.g. the Internet).
- Security and Reliable Delivery are high priority items.
- Transactions must be legally binding and auditable.
- Security and assignment of user accounts, authentication and authorization are controlled by multiple legal entities. Success depends on the close cooperation of Information Technology (IT) departments from separate companies.
- Access to ebXML services must be carefully controlled, logged and audited to guard against unauthorized access.
- Some exchanged information is sensitive and a breach of confidential information could result in legal action.
- Disputes that do occur between legal entities could result in legal action or litigation; therefore every reasonable effort must be made to preserve evidence that may be needed for non-repudiation.
- Introduction of enhancements or new systems, applications, business processes and procedures may require consensus among multiple parties. Careful coordination between multiple IT departments is essential to ensure success.

By addressing the assumptions cited above the ebXML standards provide a robust platform for wholesale electricity market exchanges and enable ISO/RTO companies to meet the financial auditability requirements outlined in [SOX] and [PCAOB-STD].

• WS-I Basic Profile

There are many Web Service specifications in existence and the first step in establishing a messaging standard is making sure all possible trading partners or parties are able to interoperate. This is the main objective of the WS-I Basic Profile.

The Web Services Basic Profile (WS-BP) is a set of recommendations on how to use Web Services specifications to maximize interoperability and is a standard developed by the Web Services Interoperability group.

The standard helps ensure that applications developed using different technologies and running on different platforms can successfully interact with each other. The standard also specifies the protocol versions that are required for compatibility.

○ Basic Profile 1.0

Version 1.0 of Basic Profile was finalized on 08/12/2003. It includes the following specifications:

- Simple Object Access Protocol (SOAP) 1.1 (http://www.w3.org/TR/SOAP/)
- Extensible Markup Language (XML) 1.0 (Second Edition) (http://www.w3.org/TR/REC-xml)
- RFC2616: Hypertext Transfer Protocol — HTTP/1.1 (http://www.ietf.org/rfc/rfc2616)
- Web Services Description Language (WSDL) 1.1 (http://www.w3.org/TR/wsdl.html)
- UDDI Version 2.04 API Specification, Dated 19 July 2002 (http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm)
- UDDI Version 2.03 Data Structure Reference, Dated 19 July 2002 (http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm)
- UDDI Version 2 XML Schema (http://uddi.org/schema/uddi_v2.xsd)
- XML Schema Part 1: Structures (http://www.w3.org/TR/xmlschema-1)
- XML Schema Part 2: Datatypes (http://www.w3.org/TR/xmlschema-2)

The list above contains the most important specifications. Basic Profile 1.0 also contains specifications regarding HTTP State Management mechanism, HTTP over SSL, the TLS protocol, SSL and PKI.

In order to conform to Basic Profile 1.0, one has to use HTTP 1.1, SOAP 1.1, WSDL 1.1, XML 1.0 and UDDI 2. Of course, the Profile allows the use of applicable extensions for any of the specified protocols.

○ Basic Profile 1.1

Version 1.1 of Basic Profile was finalized on 08/24/2004. It is an evolutionary change to Basic Profile 1.0.

Version 1.1 re-architected the profile by relocating the binding-specific envelope serialization requirements to its own profile – Simple SOAP Binding Profile 1.0. This new structure allows Basic Profile 1.1 to easily compose with any profile that specifies envelope serialization. Building on top of that, another profile, called Attachments Profile 1.0, adds support for SOAP messages with attachments.

• Message Exchange Patterns

As indicated above the WS-I Basic Profile provides the framework for Web Services to successfully interact with each other. The next step is to define a standard for how business messages will be exchanged by selecting an appropriate Message Exchange Pattern (MEP).

A MEP is a pattern for the exchange of messages between applications. The two basic ways to exchange Web Services are:

- One-way (in-only, out-only)
- Request – Response (in-out, out-in)

Application level MEPs are:

- Basic Callback – This pattern is useful for data exchanges where the data you are sending or receiving from a trading partner is specific for that trading partner.
    - The consumer calls the Web Service by sending a SOAP message bound to an HTTP request to the provider (this is the initial request)
    - The provider acknowledges receipt by sending a SOAP message bound to an HTTP response to the consumer (this is the initial response).
    - The provider then completes the exchange by sending a SOAP message bound to an HTTP request to the consumer with the results of the initial request (this is the final request, or callback).
    - The consumer acknowledges receipt of the callback message with a SOAP message bound to an HTTP response. This is the final response.

- Publish / Subscribe – this pattern is useful for data exchanges where the data exchanged is not specific to a trading partner and can be shared with multiple trading partners or parties.
    - The consumer initiates the service by sending a SOAP message bound to an HTTP request to the provider. This is the initial request which specifies what messages the consumer is subscribing to.
    - The provider acknowledges receipt by sending a SOAP message bound to an HTTP response to the consumer. This is the subscription confirmation response.
    - The provider sends the messages of the requested type to subscriber. This step repeats until the subscriber cancels the subscription.
    - The consumer cancels the subscription by sending a SOAP message bound to an HTTP request to the provider. This is the final request.
    - The provider acknowledges receipt of the cancel message with a SOAP message bound to an HTTP response. This is the final response.

• ebMS Overview

A MEP provides the pattern for how business messages will be exchanged, but this does not provide the framework for how trading partners exchange business messages in a reliable manner. An ideal reference for establishing this standard is the ebMS specification.

ebMS is one of several specifications to come from the ebXML project. In an attempt to consolidate separate business-to-business (B2B) standards initiatives the ebXML group and several vendors working on the SOAP specification agreed to collaborate to create a single B2B messaging standard using the combination of SOAP and ebMS. This crucial decision has allowed the ebMS to benefit from the popularity and familiarity of SOAP while at the same time providing auditability, reliability and security capabilities needed for robust business transactions that was lacking in SOAP.

In using ebMS you do not need to buy into the entire ebXML framework.

ebMS defines several core functions; the most important conceptually is message packaging. ebMS provides a SOAP envelope (a SOAP message with Attachments container) into which two major containers are placed, a header and a payload container. The header facilitates the functionality of the message service while the payload is the actual business message that is to be communicated to the other party.

- SOAP Header—much like a TCP/IP packet, the header contains source and destination information, timestamps, and unique identifiers. It also contains digital signatures, and a manifest describing the payload. The manifest may contain unique digital signatures for each payload item.

- Payload—an ebXML MS message may optionally include a payload. As mentioned previously, there are no specific restrictions on the type of payload that may be attached to a message. In the case of text-based payloads such as XML or EDI, these will be included in the final enveloped message in their text format. Binary files are necessarily encoded for inclusion in an ebMS message.

- Security—Digital Signature creation and verification are the primary security functions to be provided by ebMS. Additionally, encryption, authentication and authorization services may be implemented depending on availability of technology to provide these features. Each payload may have its own digital signature documented in the manifest.

- Error handling and support for Synchronous messaging are also core features.

Additional Features:

- Reliable Messaging—the most valuable of ebMS's additional features. Simply, one can designate certain (or all) messages to be sent once-and-only-once and the MSH guarantees the delivery. Messages are persisted at the sending end of a conversation to ensure message delivery even in the case of a system failure.

- Multihop Module—in the case where the final MSH is not directly accessible but there exists one or more intermediate MSH systems in between, the Multihop feature is a necessity. ebMS allows for an intermediate MSH to act as a store-and-forward facility for messages destined for some other final destination.

- Message Status Service is important for auditability purposes and allows checking on sent messages, and a Message Order Module to ensure correct message sequencing are also included as additional features of ebMS.

• Using the ebMS Message Header

SOAP is used as the transport mechanism and the body of the SOAP contains the Business Message to be exchanged. A message in general is looked at as having an envelope that contains a letter. The envelope of a message is what is referred to as being the message header and the letter as being the actual business message payload.

SOAP does not provide any quality of service capabilities or transaction support, SOAP does have an optional Header element that can be used to encapsulate extensions without having to couple them to the message payload. This allows extensions like transactions, encryption, references, and others to be added without breaking the specification.

The ebMS MessageHeader is an ideal reference model that supports the messages purpose, transaction and routing information.

Below contains the elements that make up the ebMS MessageHeader (<eb:MessageHeader>):

Note: for more details see the ebXML Message Service Specification version 2.0, section 3.1.

- **id** attribute—this MAY be used to uniquely identify this element.

- **version** attribute—is required and indicates the ebMS Header Specification and its purpose is for future versioning capabilities.

- **mustUnderstand** attribute—with a value of '1' indicates whether the contents of the MessageHeader element MUST be understood or rejected. With a value of '0' (the default), indicates the element may be ignored if not understood.

- **From** element—identifies the trading partner or party that is sending the message.
    - Contains a **PartyId** element and the value of this attribute MUST be mutually agreed and understood by each trading partner or party.
    - Contains a **Role** element that identifies the role of the trading partner or party sending the message. The value of this element is specified in the Collaboration Protocol Agreement (CPA).

      Note: the ebXML CPA is a specification that contains the technical agreement between two (or more) trading partners or parties to engage in a collaborative data exchange. For more information see the ebXML Business Process Definition Model.

- **To** element—identifies the trading partner or party that is receiving the message.
    - Contains a **PartyId** element and the value of this attribute MUST be mutually agreed and understood by each trading partner or party.

    - Contains a **Role** element that identifies the role of the trading partner or party sending the message. The value of this element is specified in the Collaboration Protocol Agreement (CPA).

- **CPAId** element—identifies the constraints that govern the data exchange between the trading partners.

- **ConversationId—**enables the receiver of the message to identify a set of related messages that make up a conversation between the two trading partners or parties. This must be unique and the party initiating the conversation determines the value. This value must remain constant for all messages within a conversation.

- **Service** element—identifies the **service** that processes the message. This is determined by the receiving trading partner.
    - This element has a single **type** attribute that MAY be used to help interpret the content of the message.

- **Action** element—identifies a process within the Service and is determined either by the receiving trading partner or by the two (or more) trading partners.

- **MessageData** element—provides the ability to uniquely identify the message. This element contains the following sub-elements:
    - **MessageId—**unique identifier for each message.
    - **Timestamp—**represents the time that the message header was created and must be expressed as UTC (or GMT).
    - **RefToMessageId—**an optional element. When it is present it MUST contain the MessageId of an earlier message that this message relates.
    - **TimeToLive—**represents the time that a message should be delivered to the trading partner and must be expressed as UTC (or GMT).

- **DuplicateElimination** element—an optional element. If present, the sender tells the receiver to check for duplicate messages.

- **Description** element—an optional element. If present, provides a description of the message.

MessageHeader Example:

```
<eb:MessageHeader eb:id="..." eb:version="2.0" SOAP:mustUnderstand="1">
       <eb:From>
              <eb:PartyId>Entity</eb:PartyId>
              <eb:Role>Managing</eb:Role>
       </eb:From>
       <eb:To>
              <eb:PartyId>TradingPartner</eb:PartyId>
              <eb:Role>Non-Managing</eb:Role>
       </eb:To>
       <eb:CPAId>http://www.rtosample.org/cpa/123456</eb:CPAId>
       <eb:ConversationId>987654321</eb:ConversationId>
       <eb:Service eb:type="myservice">ProcessData</eb:Service>
       <eb:Action>New</eb:Action>
       <eb:MessageData>
              <eb:MessageId>UUID1</eb:MessageId>
              <eb:Timestamp>2005-09-25T14:14:36</eb:Timestamp>
              <eb:RefToMessageId>UUID2</eb:RefToMessageId>
       </eb:MessageData>
       <eb:DuplicateElimination/>
</eb:MessageHeader>
```

• XML Vocabularies

Now that the messaging standards have been establishing we can start thinking about the data we will exchange or what the payload of the message will be. Utilizing a common data model or XML Vocabulary that provides a common basis of understanding would be ideal.

○ CIM

In the Energy and Utilities space the Common Information Model (CIM) which is an International Electrotechnical Commission (IEC) standard has been used for the integration of systems worldwide.

CIM is developed by and continues to be extended by WG13 and WG14 of the IEC, in collaboration with the Electric Power Research Institute (EPRI) Control Center Application Program Interface (CCAPI) Project and the Open Applications Group (OAG).

The CIM represents all the major objects in an electric utility enterprise. The model includes public classes and attributes for these classes, as well as the relationships between them. A key purpose of the CIM is to provide a common language for describing exactly what data is being exchanged among Abstract Components of Business Functions. As opposed to using custom defined tags for information fields in message payloads, fields identification is based on a class.attribute name and association relationships defined in the CIM and the OAG models.

• Conclusion

By combining the advanced messaging features of the ebMS with the interoperability mechanisms of the Basic Profile, MEPs and CIM, RTOs have been able to implement a reliable exchange for critical data utilizing the latest in Web Service technologies. This has allowed RTOs to quickly address evolving requirements by taking a standards based approach to cross-organizational information exchanges.

About the Authors

Maxime Pitard holds a Bachelor of Science in Mathematics and Computer Science from the Colorado School of Mines. He brings more than fifteen years of consulting experience to his role as a Chief Architect in the Consulting Practice of Wipro Technologies (www.wipro.com)—where he provides thought leadership on business integration solutions for global customers. His current focus of interest is on the integration of grid computing applications to Service Oriented Architectures.

Carl Ozkaynak holds a Bachelor of Science in Data Analysis from the University of New Brunswick Canada. He brings more than 10 years of experience to his role as a Senior Architect in the Consulting Practice of Wipro Technologies (www.wipro.com). Carl has designed, delivered and lead projects for several client solutions, including strategic Service-Oriented Architecture, Enterprise Application Integration, Business-to-Business, Enterprise Portal, and Web Services for a variety of diverse industries including Energy & Utilities, Financial Services, Pharmaceutical, and Telecommunications.

Dick Brooks is responsible for Enterprise Architecture at ISO New England (www.iso-ne.com), the Regional Transmission Operator (RTO) for New England. He is one of the original designers of the ebXML Message Service standard and was responsible for implementing an ebXML B2B solution for ERCOT (www.ercot.com) the Texas Independent System Operator (ISO) while employed by Systrends (www.systrends.com).

Resources

- Message Service Specification – version 2.0 - http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf
- Organization for the Advancement of Structured Information Standards – http://www.oasis-open.org
- [SOX] – http://www.pcaobus.org/About_the_PCAOB/Sarbanes_Oxley_Act_of_2002.pdf
- [PCAOB-STD] – http://www.pcaobus.org/Standards/index.aspx
- [ebXML] – http://www.ebxml.org/
- Introduction to ebXML - http://dev2dev.bea.com/pub/a/2004/12/ebXML.html
- ebXML Message Service — Implementing a Really Useful Standard - http://www.ecominfo.net/xml/arts/information_builders.htm
- XML vocabularies - http://www.service-architecture.com/xml/articles/xml_vocabularies.html
- CIM User Site - http://www.cimusers.org/
- ebXML Message Header - http://lists.ebxml.org/archives/ebxml-transport/200002/msg00006.html#1

- Cover Pages: Proposed Technical Specification for Web Services Addressing and Referencing Framework. - http://xml.coverpages.org/ni2004-06-03-a.html
- webservices.xml.com: The ebXML Messaging Service - http://webservices.xml.com/pub/a/ws/2003/03/18/ebxml.html?page=2
- Business Process Model User's Guide - http://sybooks.sybase.com/onlinebooks/group-pd/pdd1110e/bpug/@Generic__BookTextView/12038
- ebXML Business Process - http://www.gca.org/papers/xmleurope2001/papers/html/s18-1.html
- webservices.xml.com: A Preview of WS-I Basic Profile 1.1 - http://webservices.xml.com/pub/a/ws/2003/09/16/wsbp.html
- SOAP Basics - SOAP Messages - http://www.vbip.com/books/1861005091/chapter_5091_02.asp