Web Services Addressing (WS-Addressing)

13 March 2003

Authors

Adam Bosworth, BEA
Don Box, Microsoft (Editor)
Erik Christensen, Microsoft
Francisco Curbera, IBM (Editor)
Donald Ferguson, IBM
Jeffrey Frey, IBM
Chris Kaler, Microsoft
David Langworthy, Microsoft
Frank Leymann, IBM
Steve Lucco, Microsoft
Steve Millet, Microsoft
Nirmal Mukhi, IBM
Mark Nottingham, BEA
David Orchard, BEA
John Shewchuk, Microsoft
Tony Storey, IBM
Sanjiva Weerawarana, IBM

Copyright Notice

The WS-Addressing Specification may change before final release and you are cautioned against relying on the content of this specification.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the Specification or its contents without specific, written prior permission. Title to copyright in the WS-Addressing Specification will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

Abstract

WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.

Status

WS-Addressing and related specifications are provided as-is and for review and evaluation only. BEA, IBM, and Microsoft hope to solicit your contributions and suggestions in the near future. BEA, IBM, and Microsoft Corporation make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

Web Services Addressing (WS-Addressing) defines two constructs that convey information that is typically provided by transport protocols and messaging systems in an interoperable manner. These constructs normalize this underlying information into a uniform format that can be processed independently of transport or application. The two constructs are *endpoint references* and *message information headers*.

A Web service endpoint is a (referencible) entity, processor, or resource where Web service messages can be targeted. Endpoint references convey the information needed to identify/reference a Web service endpoint, and may be used in several different ways: endpoint references are suitable for conveying the information needed to access a Web service endpoint, but are also used to provide addresses for individual messages sent to and from Web services. To deal with this last usage case this specification defines a family of message information headers that allows uniform addressing of messages independent of underlying transport. These message information headers conveys end-to-end message characteristics including addressing for source and destination endpoints as well as message identity.

Both of these constructs are designed to be extensible and re-usable so that other specifications can build on and leverage endpoint references and message information headers.

The following example illustrates the use of these mechanisms in a SOAP 1.2 message being sent from http://business456.com/client1 to http://fabrikam123.com/Purchasing:

```
(001) <S:Envelope xmlns:S="http://www.w3.org/2002/12/soap-envelope"
         xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
(002)   <S:Header>
(003)     <wsa:ReplyTo>
(004)       <wsa:Address>http://business456.com/client1</wsa:Address>
(005)     </wsa:ReplyTo>
(006)     <wsa:To>http://fabrikam123.com/Purchasing</wsa:To>
(007)     <wsa:Action>http://fabrikam123.com/SubmitPO</wsa:Action>
(008)   </S:Header>
(009)   <S:Body>
(010)     ...
(011)   </S:Body>
(012) </S:Envelope>
```

Lines (002) to (008) represent the header of the SOAP message where the mechanisms defined in the specification are used. The body is represented by lines (009) to (011).

Lines (003) to (007) contain the message information header blocks. Specifically, lines (003) to (005) specify the endpoint to which replies to this message should be sent as an Endpoint Reference. Line (006) specifies the address URI of the ultimate receiver of this message. Line (007) specifies an Action URI identifying expected semantics.

1.1. Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 2.

When describing abstract data models, this specification uses the notational convention used by the XML Infoset [7]. Specifically, abstract property names always appear in square brackets (e.g., [some property]).

When describing concrete XML schemas, this specification uses the notational convention of WS-Security [15]. Specifically, each member of an element's [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).

1.2. Namespaces

This specification uses a number of namespace prefixes throughout; they are listed in **Table 1**. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see 6).

Prefix	Namespace
S	http://www.w3.org/2002/12/soap-envelope
wsa	http://schemas.xmlsoap.org/ws/2003/03/addressing
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy
xs	http://www.w3.org/2001/XMLSchema

Table 1 Prefixes and Namespaces used in this specification

WS-Addressing is defined in terms of the XML Information Set 7. WS-Addressing is conformant to the SOAP 1.2 [11][12] processing model; SOAP 1.2 is not a requirement for using the constructs defined in this specification. WS-Addressing is also designed to be able work with WSDL 1.1 [13] described services. The examples in this specification use an XML 1.0 5 representation but this is not a requirement.

All information items defined by WS-Addressing are identified by the XML namespace URI 6 "http://schemas.xmlsoap.org/ws/2003/03/addressing". A normative XML Schema 89 document can be obtained by dereferencing the XML namespace URI.

2. Endpoint References

This section defines the model and syntax of an endpoint reference.

This specification introduces a new description element type, the endpoint reference, with the intent of supporting a set of dynamic usage patterns not currently appropriately covered by WSDL 1.1 [13]. In particular, this specification intends to facilitate the following usage scenarios:

- Dynamic generation and customization of service endpoint descriptions.
- Identification and description of specific service instances that are created as the result of stateful interactions.

- Flexible and dynamic exchange of endpoint information in tightly coupled environments where communicating parties share a set of common assumptions about specific policies or protocols that are used during the interaction.

To support these scenarios, we define a lightweight and extensible mechanism to dynamically identify and describe service endpoints and instances. Because of the current limits of the WSDL 1.1 extensibility model, the WSDL 1.1 service and port elements cannot be used to cover the use cases listed above. Endpoint references logically extend the WSDL description model (e.g., portTypes, bindings, etc.), but do not replace it. Endpoint references will be used instead of WSDL <service/> elements in the following cases:

- Specific instances of a stateful service need to be identified or its instance specific configuration details need to be transmitted.
- A lightweight, self-contained description of a service endpoint needs to be communicated. In particular, this may be necessary when the details of the endpoint configuration are already shared by the communicating parties, but specific policy information needs to be added or updated, typically as a result of a dynamic configuration process.

Endpoint references compliment and do not replace the WSDL/1.1 <wsdl:service> element. We expect solutions built on WSDL/1.1 to continue to utilize a service element. Moving forward we anticipate that endpoint references and WSDL will evolve coherently. The endpoint references may link to service elements in WSDL/1.1, and support additional scenarios in which the WSDL information is not known by a party processing a message. These scenarios may include dynamic messaging or limited capability message processors.

2.1. Information Model for Endpoint References

An endpoint reference consists of the following abstract properties:

[address] : URI (mandatory)

An address URI that identifies the endpoint. This may be a network address or a logical address.

[reference properties] : xs:any (0..unbounded).

A reference may contain a number of individual properties that are required to identify the entity or resource being conveyed. Reference identification properties are element information items that are named by QName and are required to properly dispatch messages to endpoints at the endpoint side of the interaction. The interpretation of these properties (as the use of the endpoint reference in general) is dependent upon the protocol binding and data encoding used to interact with the endpoint. Section 2.3 below defines the default binding for the SOAP protocol.

[selected port type] : QName (optional)

The QName of the primary portType of the endpoint being conveyed.

[service-port] : (QName, NCName (0..1)) (optional)

This is the QName identifying the WSDL service element that contains the definition of the endpoint being conveyed. The service name provides a link to a full description of the service endpoint. An optional non-qualified name identifies the specific port in the service that corresponds to the endpoint.

[policy] : wsp:policy (0..unbounded)

A variable number of XML policy elements as described in WS-Policy [18] describing the behavior, requirements and capabilities of the endpoint. Policies may be included in an endpoint to facilitate easier processing by the consuming application, or because the policy was dynamically generated.

2.2. Endpoint Reference XML Infoset Representation

This section defines an XML Infoset-based representation for an endpoint reference as both an XML type (wsa:EndpointReferenceType) and as an XML element (<wsa:EndpointReference>).

The wsa:EndpointReferenceType type is used wherever a Web service endpoint is referenced. The following describes the contents of this type:

```
<wsa:EndpointReference>
    <wsa:Address>xs:anyURI</wsa:Address>
    <wsa:ReferenceProperties> ... </wsa:ReferenceProperties> ?
    <wsa:PortType>xs:QName</wsa:PortType> ?
    <wsa:ServiceName PortName="xs:NCName"?>xs:QName</wsa:ServiceName> ?
    <wsp:Policy/> *
</wsa:EndpointReference>
```

The following describes the attributes and elements listed in the schema overview above:

/wsa:EndpointReference
: This represents some element of type wsa:EndpointReferenceType. This example uses the predefined <wsa:EndpointReference> element, but any element of type wsa:EndpointReferenceType may be used.

/wsa:EndpointReference/wsa:Address
: This required element (of type xs:anyURI) specifies the [address] property of the endpoint reference. This address may be a logical address or identifier for the service endpoint.

/wsa:EndpointReference/wsa:ReferenceProperties/
: This optional element contains the elements that convey the [reference properties] of the reference.

/wsa:EndpointReference/wsa:ReferenceProperties/{any}
: Each child element of ReferenceProperties represents an individual [reference property].

/wsa:EndpointReference/wsa:PortType
: This optional element (of type xs:Qname) specifies the value of the [selected port type] property of the endpoint reference.

/wsa:EndpointReference/wsa:ServiceName
: This optional element (of type xs:QName) specifies the <wsdl:service> definition that contains a WSDL description of the endpoint being referenced.

/wsa:EndpointReference/wsa:ServiceName/@PortName
: This optional attribute (of type xs:NCName) specifies the name of the <wsdl:port> definition that coresponds to the endpoint being referenced.

/wsa:EndpointReference/wsp:Policy
This optional element specifies a policy that is relevant to the interaction with the endpoint.

/wsa:EndpointReference/{any}
This is an extensibility mechanism to allow additional elements to be specified.

/wsa:EndpointReference/@{any}
This is an extensibility mechanism to allow additional attributes to be specified.

The following illustrates an endpoint reference. This element references the port of type "fabrikam:InventoryPortType" at the URI "http://www.fabrikam123.com/acct".

```
<wsa:EndpointReference xmlns:wsa="..." xmlns:fabrikam="...">
    <wsa:Address>http://www.fabrikam123.com/acct</wsa:Address>
    <wsa:PortType>fabrikam:InventoryPortType</wsa:PortType>
</wsa:EndpointReference>
```

2.3. Binding Endpoint References

When a message needs to be addressed to the endpoint, the information contained in the endpoint reference is mapped to the message according to a transformation that is dependent on the protocol and data representation used to send the message. Protocol specific mappings (or bindings) will define how the information in the endpoint reference is copied to message and protocol fields. This specification defines the SOAP binding for endpoint references. This mapping MAY be explicitly replaced by other bindings (defined as WSDL bindings or as policies); however, in the absence of an applicable policy stating that a different mapping must be used, the SOAP binding defined here is assumed to apply. To ensure interoperability with a broad range of devices, all conformant implementations MUST support the SOAP binding.

The SOAP binding for endpoint references is defined by the following two rules:

- The [address] property in the endpoint reference is copied in the [destination] header field of the SOAP message.
- Each [reference property] element becomes a header block in the SOAP message. The element information item of each [reference property] (including all of its [children] and [in-scope namespaces]) is to be added as a header block in the new message.

The next example shows how the default SOAP binding for endpoint references is used to construct a message addressed to the endpoint:

```
<wsa:EndpointReference xmlns:wsa="..." xmlns:fabrikam="...">
    <wsa:Address>http://www.fabrikam123.com/acct</wsa:Address>
```

```
    <wsa:ReferenceProperties>
        <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
    </wsa:ReferenceProperties>
</wsa:EndpointReference>
```

According to the mapping rules stated before, the address value is copied in the “To” header and the “CustomerKey” element should be copied literally as a header in a SOAP message addressed to this endpoint. The SOAP message would look as follows:

```
<S:Envelope xmlns:S="http://www.w3.org/2002/12/soap-envelope"
         xmlns:fabrikam="... ">
   <S:Header>
     ...
     <wsa:To>http://www.fabrikam123.com/acct</wsa:To>
     <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
     ...
   </S:Header>
   <S:Body>
     ...
   </S:Body>
</S:Envelope>
```

3. Message Information Headers

This section defines the model and syntax of a message information header.

The message information headers collectively augment a message with the following abstract properties:

[destination] : URI (mandatory)
The address of the intended receiver of this message.

[recipient] : endpoint reference (optional)
Endpoint reference of the intended receiver of this message.

[source endpoint] : endpoint reference (optional)
Reference of the endpoint where the message originated from.

[reply endpoint] : endpoint reference (optional)
An endpoint reference that identifies the intended receiver for replies to this message. When formulating a reply message, the sender SHOULD use the contents of the [reply endpoint] of the message being replied to to formulate the reply message. If the [reply endpoint] is absent, the sender MAY use the contents of the [source endpoint] to formulate the reply message. This property may be absent if the message has no meaningful reply (e.g., is a one-way application message), or when the reply endpoint has already been

communicated to the target by other means. If the [reply endpoint] contains embedded policy, that policy must be the complete policy for that endpoint.

[fault endpoint] : endpoint reference (optional)

An endpoint reference that identifies the intended receiver for faults related to this message. When formulating a fault message, the sender SHOULD use the contents of the [fault endpoint] of the message being replied to to formulate the fault message. If the [fault endpoint] is absent, the sender MAY use the contents of the [reply endpoint] to formulate the fault message. If both the [fault endpoint] and [reply endpoint] is absent, the sender MAY use the contents of the [source endpoint] to formulate the fault message. This property may be absent if the sender cannot receive fault messages (e.g., is a one-way application message). If the [fault endpoint] contains embedded policy, that policy must be the complete policy for that endpoint.

[action] : URI (mandatory)

An identifier that uniquely (and opaquely) identifies the semantics implied by this message.

It is RECOMMENDED that value of the [action] property is a URI corresponding to an abstract WSDL construct (e.g., message, operation, port type available at the destination endpoint). In this case, the “relates to” property determines if the action is a request or response (the “direction” of the message). It is expected that at least one interoperable URI encoding scheme for computing an action element from WSDL will be defined. An action may be associated with the WSDL definition of an operation using a Policy element; the algorithm used for computing the action element URI from the WSDL definition of the endpoint may also be identified using an attached Policy element. Finally, it is possible to explicitly associate an arbitrary URI with an operation, superceding all declared encoding algorithms. This provides support to bottom-up development models.

[message id] : URI (optional)

A URI that uniquely identifies this message in time and space. No two messages may share a [message id] property. The value of this property is an opaque URI whose interpretation beyond equivalence is not defined in this specification.

[relationship] : (QName, URI) (0..unbounded)

A pair of values that indicate how this message relates to another message. The type of the relationship is identified by a QName. The related message is identified by a URI that corresponds to the related message’s [message id] property. The message identifier URI may refer to specific message, or be the following well-known URI that means “unspecified message:”

```
http://schemas.xmlsoap.org/ws/2003/03/addressing/id/unspecified
```

This specification has one predefined relationship type:

QName	Description
wsa:Response	Indicates that this a response to the message identified by the URI.

The dispatching of incoming messages is based on three message properties. The mandatory “destination” and “action” fields identify the target processing location and the verb or intent of the message. For request-response operations, the “relates to” field allows distinguishing between the request and response messages.

Due to the range of network technologies currently in wide-spread use (e.g., NAT, DHCP, firewalls), many deployments cannot assign a meaningful global URI to a given endpoint. To allow these “anonymous” endpoints to initiate message exchange patterns and receive responses, WS-Addressing defines the following well-known URI for use by endpoints that cannot have stable, resolvable URI.

```
http://schemas.xmlsoap.org/ws/2003/03/addressing/role/anonymous
```

Requests whose [reply endpoint] and/or [fault endpoint] use this address MUST provide some out-of-band mechanism for delivering responses or faults. This mechanism may be a simple request/reply transport protocol (e.g., HTTP GET or POST). This mechanism MUST NOT be used in the endpoint identifying the destination endpoint.

3.1. Message Information Headers XML Infoset Representation

The message information header blocks provide end-to-end characteristics of a message which can be easily secured as a unit. The information in these headers is immutable and not intended to be modified along the message path.

The following describes the contents of the message information header blocks:

```
<wsa:MessageID> xs:anyURI </wsa:MessageID>
<wsa:RelatesTo RelationshipType="..."?>xs:anyURI</wsa:RelatesTo>
<wsa:To>xs:anyURI</wsa:To>
<wsa:Action>xs:anyURI</wsa:Action>
<wsa:From>endpoint-reference</wsa:From>
<wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
<wsa:FaultTo>endpoint-reference</wsa:FaultTo>
<wsa:Recipient>endpoint-reference</wsa:Recipient>
```

The following describes the attributes and elements listed in the schema overview above:

/wsa:MessageID

This optional element (of type xs:anyURI) conveys the [message id] property.

/wsa:RelatesTo

This optional (repeating) element information item contributes one abstract [relationship] property value, in the form of a (URI, QName) pair. The [children] property of this element (which is of type xs:anyURI) conveys the [message id] of the related message

/wsa:RelatesTo/@RelationshipType

This optional attribute (of type xs:QName) conveys the relationship type as a QName. When absent, the implied value of this attribute is wsa:Response.

/wsa:ReplyTo

This optional element (of type wsa:EndpointReferenceType) provides the value for the [reply endpoint] property.

/wsa:From

This optional element (of type wsa:EndpointReferenceType) provides the value for the [source endpoint] property.

/wsa:FaultTo

This optional element (of type wsa:EndpointReferenceType) provides the value for the [fault endpoint] property.

/wsa:To

This required element (of type xs:anyURI) provides the value for the [destination] property.

/wsa:Action

This required element of type xs:anyURI conveys the [action] property. The [children] of this element convey the value of this property.

/wsa:Recipient

This optional element (of type wsa:EndpointReferenceType) conveys the entire endpoint reference of the recipient. Senders MAY elect to add this header as a processing hint to downstream nodes.

Messages generated in response to message containing message information header blocks SHOULD contain message information header blocks in the reply message.

The following example illustrates using message information header blocks in a SOAP 1.2 message:

```
<S:Envelope xmlns:S="http://www.w3.org/2002/12/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
  xmlns:f123="http://www.fabrikam123.com/svc53"
>
  <S:Header>
    <wsa:MessageID>uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff
    </wsa:MessageID>
    <wsa:RelatesTo>uuid:11112222-3333-4444-5555-666666666666
    </wsa:RelatesTo>
    <wsa:ReplyTo>
      <wsa:Address>http://business456.com/client1</wsa:Address>
    </wsa:ReplyTo>
    <wsa:FaultTo>
      <wsa:Address>http://business456.com/deadletters</wsa:Address>
    </wsa:FaultTo>
    <wsa:To S:mustUnderstand="1">mailto:joe@fabrikam123.com</wsa:To>
```

```
    <wsa:Action>http://fabrikam123.com/mail#Delete</wsa:Action>
  </S:Header>
  <S:Body>
    <f123:Delete>
      <maxCount>42</maxCount>
    </f123:Delete>
  </S:Body>
</S:Envelope>
```

This message would have the following property values:

[destination] The URI mailto:joe@fabrikam123.com.

[reply endpoint] The endpoint with [address] http://business456.com/client1.

[fault endpoint] The endpoint with [address] http://business456.com/deadletters.

[action] http://fabrikam123.com/mail#Delete

[message id] uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff

[relationship] (wsa:Response,uuid:11112222-3333-4444-5555-666666666666)

4. Security Considerations

It is strongly RECOMMENDED that the communication between services be secured using the mechanisms described in WS-Security [15]. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, the message information headers described in this specification (e.g. <wsa:To>) need to be signed with the body in order to "bind" the two together. It should be noted that for messages traveling through intermediaries, it is possible that some or all of the message information headers MAY have multiple signatures when the message arrives at the ultimate receiver. It is strongly RECOMMENDED that the initial sender include a signature to prevent any spoofing by intermediaries.

Whenever an address is specified (e.g. <wsa:From>, <wsa:ReplyTo>, <wsa:FaultTo>, …), the processor SHOULD validate that a signature is provided with claims allowing it to speak for the specified target in order to prevent certain classes of attacks.

The message information headers blocks MAY have their contents encrypted in order to obtain end-to-end privacy, but care should be taken to ensure that intermediary processors have access to required information (e.g. <wsa:To>).

In some cases, intermediaries MAY add additional message information headers. If duplicate headers are present, they SHOULD be targeted at different SOAP actors/roles. If multiple headers are specified for the same actor/role, and have conflicting information, processors SHOULD favor the information from the initial sender (or its delegate) unless the processor has additional information allowing it to make a correct determination and avoid obvious security attacks.

Some processors MAY cache message identifiers (<wsa:MessageID>) in order to detect replays of messages.

The following list summarizes common classes of attacks that apply to the mechanisms in this specification and identifies the mechanism to prevent/mitigate the attacks:

- Message alteration – Alteration is prevented by including signatures of the message information using WS-Security.
- Message disclosure – Confidentiality is preserved by encrypting sensitive data using WS-Security.
- Address spoofing – Address spoofing is prevented by ensuring that all address are signed by a party authorized to speak for (or on behalf of) the address.
- Key integrity – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see WS-Policy [18] and WS-SecurityPolicy [16]).
- Authentication – Authentication is established using the mechanisms described in WS-Security and WS-Trust [17]. Each message is authenticated using the mechanisms described in WS-Security.
- Accountability – Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.
- Availability – All reliable messaging services are subject to a variety of availability attacks. Replay detection is a common attack and it is RECOMMENDED that this be addressed by the mechanisms described in WS-Security and/or caching of message identifiers. Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification.

5. Acknowledgements

Michael Coulson, Microsoft; Giovanni Della-Libera, Microsoft; Christopher Ferris, IBM; Tom Freund, IBM; Steve Graham, IBM; Maryann Hondo, IBM; Efim Hudis, Microsoft; John Ibbotson, IBM; Gopal Kakivaya, Microsoft; Al Lee, Microsoft; Anthony Nadalin, IBM; Martin Nally, IBM; Henrik Frystyk Nielsen, Microsoft; Jeffrey Schlimmer, Microsoft; Keith Stobie, Microsoft

6. References

1. B. Carpenter, Y. Rekhter, "Renumbering Needs Work", RFC 1900, IAB, February 1996
2. S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997
3. T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998
4. R. Hinden, B. Carpenter, L. Masinter, "Format for Literal Ipv6 Addresses in URL's", RFC 2732, Nokia, IBM, AT&T, December 1999
5. W3C Recommendation "Extensible Markup Language (XML) 1.0 (Second Edition)", Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, 6 October 2000 (see "http://www.w3.org/TR/2000/REC-xml-20001006)

6. W3C Recommendation "Namespaces in XML", Tim Bray, Dave Hollander, Andrew Layman, 14 January 1999 (see "http://www.w3.org/TR/1999/REC-xml-names-19990114/")
7. W3C Recommendation "XML Information Set", John Cowan, Richard Tobin, 24 October 2001 (see "http://www.w3.org/TR/2001/REC-xml-infoset-20011024/")
8. W3C Recommendation "XML Schema Part 1: Structures", Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, 2 May 2001 (see "http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/")
9. W3C Recommendation "XML Schema Part 2: Datatypes", Paul V. Biron, Ashok Malhotra, 2 May 2001 (see "http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/")
10. W3C Recommendation "XML Base", Jonathan Marsh, 27 June 2001 (see "http://www.w3.org/TR/2001/REC-xmlbase-20010627/")
11. W3C Working Draft "SOAP Version 1.2 Part 1: Messaging Framework", Martin Gudgin, Marc Hadley, Jean-Jacques Moreau, Henrik Frystyk Nielsen (see "http://www.w3.org/TR/2002/WD-soap12-part1-20020626/"). This is work in progress.
12. M. Baker, M. Nottingham, "The 'application/soap+xml' media type", Internet-Draft draft-baker-soap-media-reg-01, June 2002. This is work in progress.
13. “Web Services Description Language 1.1”, Ariba, IBM, Microsoft, February 2001 (available at “http://www.w3.org/TR/wsdl”)
14. “Business Process Execution Language for Web Services”, BEA, IBM, Microsoft, August 2002 (available at “http://msdn.microsoft.com/library/en-us/dnbiz2k2/html/bpel1-0.asp”)
15. "Web Services Security Language", IBM, Microsoft, VeriSign, April 2002.
16. “Web Services Security Policy Language”, IBM, Microsoft, RSA Security, VeriSign, December 2002.
17. “Web Services Trust Language”, IBM, Microsoft, RSA Security, VeriSign, December 2002.
18. “Web Services Policy Framework”, BEA, IBM, Microsoft, SAP, December 2002.