Basic Profile Version 1.0

Working Group Draft

Date: 2002/10/08

This version:
http://www.ws-i.org/Profiles/Basic/2002-10/BasicProfile-1.0-WGD.htm
Latest version:
http://www.ws-i.org/Profiles/Basic/2002-10/BasicProfile-1.0-WGD.htm

Editors:
Keith Ballinger, Microsoft
David Ehnebuske, IBM
Martin Gudgin, Microsoft
Mark Nottingham, BEA Systems
Prasad Yendluri , webMethods

Notice

The material contained herein is not a license, either expressly or impliedly, to any intellectual property owned or controlled by any of the authors or developers of this material or WS-I. The material contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and WS-I hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence. ALSO, THERE IS NO WARRANTY OR

NOTE: This is not a final document. This is an interim draft published for early review and comment. Some or all of this document is likely to change before final approval and publication. This document has not been approved as final Material by the WS-I membership.

CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR WS-I BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Feedback

The Web Services-Interoperability Organization (WS-I) would like to receive input, suggestions and other feedback ("Feedback") on this work from a wide variety of industry participants to improve its quality over time.

By sending email, or otherwise communicating with WS-I, you (on behalf of yourself if you are an individual, and your company if you are providing Feedback on behalf of the company) will be deemed to have granted to WS-I,the members of WS-I, and other parties that have access to your Feedback, a non-exclusive, non-transferable, worldwide, perpetual, irrevocable, royalty-free license to use, disclose, copy, license, modify, sublicense or otherwise distribute and exploit in any manner whatsoever the Feedback you provide regarding the work. You acknowledge that you have no expectation of confidentiality with respect to any Feedback you provide. You represent and warrant that you have rights to provide this Feedback, and if you are providing Feedback on behalf of a company, you represent and warrant that you have the rights to provide Feedback on behalf of your company. You also acknowledge that WS-I is not required to review, discuss, use, consider or in any way incorporate your Feedback into future versions of its work. If WS-I does incorporate some or all of your Feedback in a future version of the work, it may, but is not obligated to include your name (or, if you are identified as acting on behalf of your company, the name of your company) on a list of contributors to the work. If the foregoing is not acceptable to you and any company on whose behalf you are acting, please do not provide any Feedback.

Feedback on this document should be directed to wsbasic_comment@ws-i.org.

NOTE: This is not a final document. This is an interim draft published for early review and comment. Some or all of this document is likely to change before final approval and publication. This document has not been approved as final Material by the WS-I membership.

Abstract

This document defines the WS-I Basic Profile, consisting of a set of non-proprietary Web services specifications, along with clarifications to those specifications which promote interoperability.

Status of this Document

This document is a Working Group Draft; it has been accepted by the Working Group as reflecting the current state of discussions. It is a work in progress, and should not be considered authoritative or final; other documents may supercede this document.

Table of Contents

1. Introduction

This document defines the WS-I Basic Profile, consisting of a set of non-proprietary Web services specifications, along with clarifications to those specifications which promote interoperability.

Section 2, "Scope of the Profile," catalogues the specifications included in the profile, along with their associated functions. Section 3, "Profile Conformance," explains what it means to be conformant to the Basic Profile. Each subsequent section addresses a component specification of the Profile, and consists of two parts; an overview of the approach to the specification taken, followed by subsections which address individual parts of the component specification.

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 RFC2119.

This specification uses a number of namespace prefixes throughout; they are listed below. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

- **soap** - http://schemas.xmlsoap.org/soap/envelope/
- **xsi** - http://www.w3.org/1999/XMLSchema-instance
- **xsd** - http://www.w3.org/1999/XMLSchema
- **soapenc** - http://schemas.xmlsoap.org/soap/encoding/
- **wsdl** - http://schemas.xmlsoap.org/wsdl/
- **soapbind** - http://schemas.xmlsoap.org/wsdl/soap/

2. Scope of the Profile

A number of component technologies are used to compose a Web service. The Basic Profile dictates how a selected set of specified Web services technologies should be used together in an interoperable manner. They are:

- **Messaging** - the exchange of protocol elements, usually over a network, to effect a Web service.
- **Description** - the enumeration of the messages associated with a Web service, along with implementation details.
- **Discovery** - metadata that enables the advertisement of a Web service's capabilities.
- **Security** - mechanisms that provide integrity, privacy, authentication and authorization functions.

The profile mandates the use of a particular technology (or technologies), when appropriate, for each of these components.

Editors' note: The editors are considering placing a complete listing of incorporated specifications here.

Editors' note: The Working Group is currently considering whether to include an attachments mechanism in the Basic Profile; if so, it should be referenced here, and it may impact current requirements (e.g., allowed bindings in WSDL, allowed Content-Type values in HTTP).

3. Profile Conformance

Conformance to the Basic Profile is defined by adherence to the specifications on which the profile is based (as outlined in the remainder of the document), subject to the refinements, interpretations, and clarifications set forth.

To allow the description of conformance in different contexts, the profile defines a number of conformance targets, allowing the conformance testing and certification of artifacts (such as SOAP messages and WSDL descriptions), Web services themselves, and software that is used in conjunction with a conformant Web Service.

The criteria for conformance is defined by requirement statements, which are associated with conformance targets (denoted with capital letters, e.g., MESSAGE) and use requirement levels (using RFC2119 language, e.g., MUST) to indicate the nature of the requirement. Requirement statements are individually identified (e.g., r999) for convenience. Additional text may be included to illuminate the requirements (e.g., rationale and examples); however, requirement statements alone should be considered in determining conformance.

The sections below describe this profile's conformance targets, from the basic artifacts (upon which requirements are directly placed) to the conformance of services and software, which is derived from these artifacts and additional requirements.

3.1 Conformance of Artifacts

The most basic level of conformance is that of an artifact. This profile makes requirement statements about three kinds of artifacts;

- **MESSAGEs** - protocol elements that are exchanged, usually over a network, to effect a Web service (i.e., SOAP/HTTP messages)

- **DESCRIPTIONs** - descriptions of types, messages, interfaces and their concrete protocol and data format bindings, and the network access points associated with Web services (i.e., WSDL descriptions)"
- **METADATA** - statements about Web services that are used to discover their capabilities (i.e., UDDI tModels)

An instance of an artifact is considered conformant when all of the requirements associated with it are met.

Editors' note: The Description section of the draft currently uses PUBLISHER, not METADATA; the editors are currently considering what the most appropriate term is.

3.2 Conformance of Services

A deployed instance of a Web service (as specified by wsdl:port) is considered conformant if it produces only conformant artifacts, and is capable of consuming conformant artifacts, as appropriate. Note that this means that where multiple conformant artifacts are possible, a conformant service must be able to consume them all (e.g., while a sender may choose whether to encode XML in UTF-8 or UTF-16 when sending a message, a receiver must be capable of using either).

A Web service instance must be described by a WSDL 1.1 service description. If an authorized consumer requests a service description of a conformant service instance as a WSDL 1.1 document, then the service instance provider must make the WSDL document available to that consumer. A service instance may provide run-time access to WSDL documents from a server, but is not required to do so for WS-I Basic profile conformance.

In addition, conformant Web services must comply with all of the requirement statements associated with:

- **PROCESSORs** - software that consumes messages according to the protocol associated with them (i.e., SOAP processors)
- **INSTANCEs** - deployed instances of Web services (as specified by wsdl:port)

Types of Web services (as specified by wsdl:binding and wsdl:portType) are considered conformant if, when deployed with due consideration, they produce conformant instances.

Editors' note: The editors expect that the specification of the conformance annotation for WSDL and/or UDDI will be placed in this section, or a separate subsection of "Conformance".

4. Messaging

This portion of the profile incorporates the following specifications by reference;

- Simple Object Access Protocol (SOAP) 1.1 .
- Extensible Markup Language (XML) 1.0 (Second Edition).
- RFC2616: Hypertext Transfer Protocol -- HTTP/1.1.
- RFC2965: HTTP State Management Mechanism.

4.1 XML Representation of SOAP Messages

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- SOAP 1.1, Section 4.

SOAP/1.1 defines an XML-based structure for transmitting messages. This profile mandates the use of that structure, and places the following constraints on its use:

R4001 *MESSAGEs MAY include the Unicode Byte Order Mark (BOM).*

The XML specification allows UTF-8 encoding to include a BOM; therefore, receivers of messages must be prepared to accept them.

R1000 *When a MESSAGE contains a* `soap:Fault` *element, that element MUST NOT have element children other than* `faultcode`, `faultstring`, `faultactor` *and* `detail`.

For interoperability the content of the `soap:Fault` element is fixed.

INCORRECT:

```
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <faultcode>soap:Client</faultcode>
  <faultstring>Invalid message format</faultstring>
  <faultactor>http://example.org/someactor</faultactor>
  <detail>There were <b>lots</b> of elements in the message
  I did not understand
  </detail>
  <m:Exception xmlns:m='http://example.org/faults/exceptions' >
    <m:ExceptionType>Severe</m:ExceptionType>
  </m:Exception>
</soap:Fault>
```

CORRECT:

```
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
```

```
  <faultcode>soap:Client</faultcode>
  <faultstring>Invalid message format</faultstring>
  <faultactor>http://example.org/someactor</faultactor>
  <detail>There were <b>lots</b> of elements in the message
   I did not understand
   </detail>
</soap:Fault>
```

R1001 *When a MESSAGE contains a `soap:Fault` element its element children MUST be unqualified.*

The children of the `soap:Fault` element are local to that element and do not need to be namespace qualified.

INCORRECT:

```
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <soap:faultcode>soap:Client</soap:faultcode>
  <soap:faultstring>Invalid message format</soap:faultstring>
  <soap:faultactor>http://example.org/someactor</soap:faultactor>
  <soap:detail>There were <b>lots</b> of elements in the message I did
  not understand
  </soap:detail>
</soap:Fault>
```

CORRECT:

```
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
                       xmlns='' >
  <faultcode>soap:Client</faultcode>
  <faultstring>Invalid message format</faultstring>
  <faultactor>http://example.org/someactor</faultactor>
  <detail>There were <b>lots</b> of elements in the message I did
  not understand
  </detail>
</soap:Fault>
```

R1002 *The `detail` element of a SOAP Fault MESSAGE MAY have any elements from any namespace, including qualified elements as children.*

R1003 *The `detail` element of a SOAP Fault MESSAGE MAY have any qualified attribute whose [namespace name] is NOT "http://schemas.xmlsoap.org/soap/envelope/".*

For extensibility, both attributes and elements are allowed.

R1004 *When a MESSAGE contains a `faultcode` element the content of that element MUST be one of the fault codes defined in the SOAP 1.1 specification. Custom fault codes MUST NOT appear inside the `faultcode` element.*

For interoperability a fixed set of fault codes is needed.

INCORRECT:

```
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
            xmlns:c='http://example.org/faultcodes' >
  <faultcode>c:ProcessingError</faultcode>
  <faultstring>An error occured while processing the message
  </faultstring>
</soap:Fault>
```

CORRECT:

```
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <faultcode>soap:Server</faultcode>
  <faultstring>An error occured while processing the message
  </faultstring>
</soap:Fault>
```

R1005 *MESSAGEs MUST NOT contain `soap:encodingStyle` attributes on any of the elements whose [namespace name] is "http://schemas.xmlsoap.org/soap/envelope/".*

R1006 *MESSAGEs MUST NOT contain `soap:encodingStyle` attributes on any element which is a child of `soap:Body`.*

R1007 *MESSAGEs MUST NOT contain `soap:encodingStyle` attributes on any elements which are grandchildren of `soap:Body`.*

For interoperability, literal XML is preferred.

R1008 *A MESSAGE MUST NOT contain a Document Type Declaration.*

R1009 *A MESSAGE MUST NOT contain Processing Instructions.*

For interoperability and ease of processing these XML constructs are disallowed.

R1010 *A MESSAGE MAY contain an XML Declaration.*

Presence or absence of such a declaration does not affect interoperability. Certain implementations might always precede their XML serialization with the XML declaration.

R1011 *A MESSAGE MUST NOT have any element children of* `soap:Envelope` *following the* `soap:Body` *element.*

INCORRECT:

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <soap:Body>
    <p:Process xmlns:p='http://example.org/Operations' />
  </soap:Body>
  <m:Data xmlns:m='http://example.org/information' >
  Here is some data with the message
  </m:Data>
</soap:Envelope>
```

CORRECT:

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <soap:Body>
    <p:Process xmlns:p='http://example.org/Operations' >
         <m:Data xmlns:m='http://example.org/information' >
  Here is some data with the message
      </m:Data>
    </p:Process>
  </soap:Body>
</soap:Envelope>
```

R1012 *MESSAGEs MAY be serialized as either UTF-8 or UTF-16.*

All XML Processors must support UTF-8 and UTF-16, per the XML 1.0 specification.

R1013 *MESSAGEs containing a* `mustUnderstand` *attribute MAY use any of the four lexical values (0, 1, false, true) as a value.*

The `mustUnderstand` attribute has a type of xs:boolean which allows all four lexical forms.

R1014 *The children of the* `soap:Body` *element in a MESSAGE MUST be namespace qualified.*

The interpretation of unqualified is ambiguous, therefore qualified names must be used.

R1015 *PROCESSORs MUST generate a fault if they encounter a message whose document element has a local name of "Envelope" but a namespace name which is not "http://schemas.xmlsoap.org/soap/envelope/".*

SOAP 1.1 only stated that the message be discarded in such cases. For interoperability faults must be generated instead.

R1016 *When a MESSAGE contains a* `soap:Fault` *element, the* `faultstring` *element child MAY carry an* `xml:lang` *attribute.*

R1017 *A PROCESSOR MUST NOT mandate the use of the* `xsi:type` *attribute in messages except as required in order to indicate a derived type (see XML Schema Part 1: Structures, Section 2.6.1).*

4.2 The SOAP Processing Model

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- SOAP 1.1, Section 2.

SOAP/1.1 defines a message exchange model for processing of messages. This profile places the following constraints on that model:

R1025 *PROCESSORs MUST handle messages in such a way that it appears that all checking of mandatory headers is performed before any actual processing.*

This guarantees that no undesirable side-effects will occur as a result of noticing a mandatory header AFTER processing other parts of the message.

R1026 *The value of the* `soap:actor` *attribute in a MESSAGE is a private agreement between the sender and the receiver of the header carrying the attribute.*

Editors' note: This statement isn't really a requirement; it might become a Best Practice.

R1027 *PROCESSORs MUST generate a mustUnderstand fault when a message contains a mandatory header that the processor does not understand. A mandatory header is one which carries a* `mustUnderstand` *attribute with the value 1 or true.*

This ensures that mandatory headers are not silently and erroneously ignored.

R1028 *Upon generating a SOAP Fault a PROCESSOR MUST NOT effect any further processing of a SOAP message beyond that which is necessary to handle the generated SOAP Fault.*

R1029 *Where the normal outcome of processing a SOAP message would have resulted in the transmission of a SOAP response, but rather a SOAP Fault is generated instead, a PROCESSOR MUST transmit a SOAP Fault message in place of the response.*

R1030 *A PROCESSOR that generates a SOAP Fault SHOULD notify the end user that a SOAP Fault has been generated when practical, by whatever means is deemed appropriate to the circumstance.*

These requirements ensure that, when a Fault is generated, no further processing will be done on the message, a Fault message will be transmitted to the sender of the request message in request-response cases and some application level error will be flagged to the user.

4.3 Using SOAP in HTTP

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- SOAP 1.1, Section 6.
- HTTP/1.1.
- HTTP State Management Mechanism.

SOAP/1.1 defines a single protocol binding, for HTTP. This profile mandates the use of that binding, and places the following constraints on its use:

R1140 *MESSAGEs SHOULD be sent using HTTP/1.1.*

HTTP/1.1 has several performance advantages and is more clearly specified, in comparison to HTTP/1.0. Note that support for HTTP/1.0 is implied in HTTP/1.1, and that intermediaries may change the version of a message; for more information about HTTP versioning, see RFC2145.

R1106 *A MESSAGE identified as a Fault MUST use the "500 Server Error" HTTP status code in the HTTP binding.*

R1107 *PROCESSORs MUST interpret SOAP messages containing only a* `soap:Fault` *element as a Fault.*

Some processor implementations use only the HTTP status code to determine the presence of a SOAP Fault. Because there are situations where the Web infrastructure changes the

HTTP status code, and for general reliability, the Profile requires that they examine the envelope.

R1108 *MESSAGEs MUST NOT use the HTTP Extension Framework [RFC2774].*

The HTTP Extension Framework is an experimental mechanism for extending HTTP in a modular fashion. Because it is not deployed widely and also because the benefits to the use of SOAP are questionable, the Profile does not allow its use.

R1109 *If a MESSAGE carries a* `SOAPAction` *HTTP header that header MAY contain any quoted string including "".*

The `SOAPAction` header is purely a hint to processors. All vital information regarding the intent of a message is carried in the Envelope.

R1110 *INSTANCEs MAY use TCP port 80 (HTTP).*

SOAP is designed to take advantage of the HTTP infrastructure. However, there are some situations (e.g., involving proxies, firewalls and other intermediaries) where there may be harmful side effects. As a result, instances may find it advisable use other ports.

R1111 *INSTANCEs SHOULD respond to a request-response based request message with a "200 OK" HTTP status code if the response contains a SOAP message which is not a SOAP Fault.*

R1112 *INSTANCEs SHOULD respond to a one-way request message with a "202 Accepted" HTTP status code if no SOAP Fault is generated.*

R1113 *INSTANCEs SHOULD respond to a request message with a "400 Bad Request" HTTP status code if the request payload is malformed.*

R1114 *INSTANCEs SHOULD respond to a request message with a "405 Method not Allowed" HTTP status code if the request method was not "POST".*

R1115 *INSTANCEs SHOULD respond to a request message with a "415 Unsupported Media Type" HTTP status code if the* `Content-Type` *HTTP request header did not have a value of "text/xml".*

R1116 *INSTANCEs SHOULD respond to a request message with a "500 Internal Server Error" HTTP status code if the response message contains a SOAP Fault.*

Consistent use of HTTP status codes is vital to interoperability between consumers and services.

R1120 *INSTANCEs MAY attempt to use the HTTP state mechanism ("cookies").*

R1121 *INSTANCES MUST NOT require support for cookies in order to function correctly.*

R1122 *If INSTANCEs use the HTTP state mechanism, they SHOULD conform to that described in RFC2965.*

HTTP cookies are a useful tool for improving the efficiency of a service (e.g., through session management). However, cookie support in clients is not mandated by RFC2965, and therefore cannot be required for successful operation; it should only be used as an optimization or hint.

R1130 *INSTANCEs MUST use HTTP status code "307 Temporary Redirect" when redirecting a request to a different endpoint.*

There are interoperability problems with using many of the HTTP redirect status codes, generally surrounding whether to use the original method, or GET. The profile mandates "307 Temporary Redirect" as the correct status code for redirection; for more information, see the 3xx status code descriptions in RFC2616.

5. Service Description

The profile uses Web Services Description Language (WSDL) to enable the description of services as a set of endpoints operating on messages.

This portion of the profile incorporates the following specifications by reference;

- Web Services Description Language (WSDL) 1.1.
- XML Schema Part 1: Structures.
- XML Schema Part 2: Datatypes.

5.1 Document Structure

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 2.1.

WSDL/1.1 defines an XML-based structure for describing Web services. This profile mandates the use of that structure, and places the following constraints on its use:

R2001 *A DESCRIPTION MUST only use the WSDL "import" statement to import another WSDL description.*

R2002 *A DESCRIPTION MUST use the XML Schema "import" statement to import XML Schema Definitions.*

R2003 *A DESCRIPTION MUST only use the XML Schema "import" statement within the schema element of the types section.*

R2004 *A DESCRIPTION SHALL NOT use the XML Schema "import" statement to import a Schema definition embedded within another WSDL description.*

To promote interoperability the import mechanisms are kept consistent and confined to their respective domains; the wsdl related mechanism in the "wsdl" domain and the schema related mechanisms in "schema" domain where the normal rules from the schema specification can be applied consistently.

INCORRECT:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/definitions"
   xmlns:xsd1="http://example.com/stockquote.xsd"
                ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/schemas"
         location="http://example.com/stockquote/stockquote.xsd"/>

   <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
               ...
</definitions>
```

CORRECT:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/definitions"
   xmlns:xsd1="http://example.com/stockquote.xsd"
                ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <types>
     <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <xsd:import namespace="http://example.com/stockquote/schemas"
         schemaLocation="http://example.com/stockquote/stockquote.xsd"/>
     </xsd:schema>
   </types>
```

```
   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>
               ...
</definitions>
```

CORRECT:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/definitions"
   xmlns:xsd1="http://example.com/stockquote.xsd"
               ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/definitions"
        location="http://example.com/stockquote/stockquote.wsdl"/>

   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>
               ...
</definitions>
```

R4002 *DESCRIPTIONs MAY include the Unicode Byte Order Mark (BOM)*

The XML specification allows UTF-8 encoding to include a BOM; therefore, description processors must be prepared to accept them.

R2005 *The value of the* `targetNamespace` *attribute on the* `wsdl:definitions` *element of a DESCRIPTION that is being imported MUST match the value given to the* `namespace` *attribute on the* `wsdl:import` *element in the importing DESCRIPTION , that references the imported DESCRIPTION .*

R2007 *A DESCRIPTION MUST specify a value for the* `location` *attribute on the* `wsdl:import` *element.*

R2008 *The value of the* `location` *attribute of a* `wsdl:import` *element MUST be treated as a hint.*

R2020 *The* `wsdl:documentation` *element MAY occur under the* `wsdl:import` *element in a DESCRIPTION.*

R2021 *The `wsdl:documentation` element MAY occur under the `wsdl:part` element in a DESCRIPTION.*

Eliminate inconsistency between WSDL schema and the WSDL specification in this area.

R2022 *In a DESCRIPTION the `wsdl:types` element MUST occur either as the first child of the `wsdl:definitions` element if no `wsdl:import` element is present; or immediately following the `wsdl:import` element(s) if present.*

R2023 *In a DESCRIPTION the `wsdl:import` element(s), when present, MUST occur prior to any other child elements under the `wsdl:definitions` element.*

Eliminate confusion created by example 3 in section 3.1 of the WSDL 1.1 specification and also align with the W3C WSDL WG resolution on this.

INCORRECT:

```
<definitions name="StockQuote"
                ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/definitions"
         location="http://example.com/stockquote/stockquote.wsdl"/>

   <message name="GetLastTradePriceInput">
       <part name="body" type="tns:TradePriceRequest"/>
   </message>
              ...
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           ....
      </port>
   </service>

   <types>
      <schema targetNamespace="http://example.com/stockquote.xsd"
               xmlns="http://www.w3.org/2000/10/XMLSchema">
           .......
      </schema>
   </types>
```

```
</definitions>
```

CORRECT:

```
<definitions name="StockQuote"
                ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import namespace="http://example.com/stockquote/definitions"
          location="http://example.com/stockquote/stockquote.wsdl"/>

  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
         xmlns="http://www.w3.org/2000/10/XMLSchema">
           .......
    </schema>
   </types>

   <message name="GetLastTradePriceInput">
      <part name="body" element="tns:TradePriceRequest"/>
   </message>
              ...
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           ....
      </port>
   </service>
</definitions>
```

CORRECT:

```
<definitions name="StockQuote"
                ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
     <schema targetNamespace="http://example.com/stockquote.xsd"
```

```
          xmlns="http://www.w3.org/2000/10/XMLSchema">
           .......
     </schema>
   </types>

   <message name="GetLastTradePriceInput">
        <part name="body" element="tns:TradePriceRequest"/>
   </message>
               ...
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           ....
      </port>
   </service>
</definitions>
```

5.2 Types

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 2.2.

The `wsdl:types` element of WSDL/1.1 encloses data type definitions that are relevant to the Web service described. This profile places the following constraints pertinent to the `wsdl:types` element:

R2101 *A DESCRIPTION MUST NOT use QName references to things in namespaces that have not been imported.*

R2110 *A DESCRIPTION MUST NOT use `soapenc:arrayType` attribute.*

5.3 Messages

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 2.3.

In WSDL/1.1 message elements are used to represent abstract definitions of the data being transmitted. This profile places the following constraints on the use of the message element(s):

R2201 *If* `style`*="document" and* `use`*="literal" at the SOAP binding level, a DESCRIPTION MUST have zero or one part in a* `wsdl:message` *element that forms the* `soap:body`*.*

R2202 *If* `style`*="rpc" and* `use`*="literal" at the SOAP binding level, a DESCRIPTION MAY have zero parts in a* `wsdl:message` *element that forms the* `soap:body`*.*

R2203 *If* `style`*="rpc" and* `use`*="literal" at the SOAP binding level, a DESCRIPTION MUST use the* `type` *attribute to define the part(s) in a* `wsdl:message` *element.*

R2204 *If the* `style`*="document" and* `use`*="literal" at the SOAP binding level, a DESCRIPTION MUST use the* `element` *attribute to define the single part in a message.*

Use of `wsdl:message` elements with zero parts is permitted in both RPC and Document styles to permit operations that can send or receive MESSAGEs with empty SOAP Bodies. This case is explicitly permitted by the Basic Profile.

R2205 *In a DESCRIPTION, when the* `element` *attribute is used to define a part in a* `wsdl:message` *element, the value of the* `element` *attribute MUST refer to an element definition.*

The examples 4,5 in section 3.1 of the WSDL 1.1 specification incorrectly show the use of Schema types (e.g. xsd:string) as a valid value for the `element` attribute of a `wsdl:part` element.

INCORRECT:

```
<message name="GetTradePriceInput">
    <part name="tickerSymbol" element="xsd:string"/>
    <part name="time" element="xsd:timeInstant"/>
</message>

<message name="GetTradePricesInput">
    <part name="tickerSymbol" element="xsd:string"/>
</message>
```

CORRECT:

```
<message name="GetTradePriceInput">
    <part name="body" element="tns:SubscribeToQuotes"/>
</message>
```

5.4 Port Types

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 2.4.

In WSDL/1.1, `portType` elements are used to group a set of abstract operations. This profile places the following constraints on the use of `portType` element(s):

R2301 *The order of the parts in a message in the DESCRIPTION MUST be the definitive order of the part elements on the wire for any part in an operation.*

R2302 *A DESCRIPTION MAY use the `parameterOrder` attribute of an `wsdl:operation` element to indicate the return value and method signatures as a hint to code generators.*

Permitting the use of parameterOrder helps code generators in mapping between method signatures and on the wire MESSAGE instances.

R2303 *A DESCRIPTION MUST NOT use Solicit-Response and Notification type operations.*

Solicit-Response and Notification are not well defined by the WSDL/1.1 specification and the WSDL/1.1 specification defines bindings for the One-way and Request-response primitives only.

R2304 *All operations within a portType in a DESCRIPTION MUST have distinct values for the `name` attribute.*

To promote interoperability operation overloading is disallowed by the Basic Profile.

R2305 *In a DESCRIPTION operations within a portType that represent RPC style functions MUST have 0 or 1 part in the `wsdl:message` element that represents a return value. The single part can however represent instance of a complex type.*

Having more than 1 part in return value is not meaningful for RPC style operations.

5.5 Bindings

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 2.5.

In WSDL/1.1 the `binding` element supplies the concrete protocol and data format specifications for the operations and messages defined by a particular portType. This profile places the following constraints on the binding specifications:

> R2401 *A DESCRIPTION MUST use WSDL SOAP Binding that is defined in section "3 SOAP Binding" of the WSDL 1.1 specification.*

For interoperability the choice of bindings is limited to the well defined and most commonly used one.

5.6 Ports

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 2.6.

In WSDL/1.1 the port element specifies an address for binding on a portType, thus defining a communication end-point for the Web service. This profile places the following constraints on the use of the port element:

> **Editors' note:**The Working Group has not closed any issues relating to Ports as of publication.

5.7 Services

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 2.7.

In WSDL/1.1 the service element is used to aggregate a set of related ports. This profile places the following constraints on the use of the service element:

> **Editors' note:**The Working Group has not closed any issues relating to Services as of publication.

5.8 SOAP Binding

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- WSDL 1.1, Section 3.0.

WSDL/1.1 defines a binding for SOAP 1.1 endpoints. This profile mandates the use of SOAP binding as defined in the WSDL/1.1 specification, and places the following constraints on its use:

R2700 *A DESCRIPTION MUST use SOAP 1.1 protocol with SOAP Binding.*

SOAP 1.2 specification differs from the SOAP 1.1 specification in many respects. For interoperability the profile limits the SOAP binding to the SOAP 1.1 protocol.

R2701 *A DESCRIPTION MUST have the* `transport` *attribute of the* `soapbind:binding` *element specified in the SOAP binding description.*

Eliminate inconsistency between the WSDL 1.1 specification text and the WSDL 1.1 schema. The WSDL 1.1 specification shows it to be required but, the schema shows this attribute to be optional, where as the Basic profile sees this to be a required attribute.

R2702 *A DESCRIPTION MUST use HTTP transport protocol with SOAP binding. Specifically, the* `transport` *attribute of* `soapbind:binding` *element MUST have the value "http://schemas.xmlsoap.org/soap/http".*

For interoperability the transport protocol is limited to HTTP. To permit secure transfers at the HTTP level use of HTTP(S) is allowed.

R2705 *A DESCRIPTION MUST use the same value of, either "rpc" or "document" for the* `style` *attribute for all of its operations in a portType, in the SOAP Binding description.*

Disallow mix and match of operation "style" in the same port.

R2706 *A DESCRIPTION MUST use the value of "literal" for the* `use` *attribute in the SOAP Binding description.*

R2707 *If a DESCRIPTION does not specify the* `use` *attribute in the SOAP Binding description, the value of the* `use` *attribute SHALL default to the value "literal".*

For interoperability the profile prohibits the use of different encodings including the SOAP encoding.

R2708 *A DESCRIPTION MUST have at least one profile compliant binding per portType.*

R2709 *A DESCRIPTION MAY have more than one profile compliant bindings per portType.*

R2710 *The Basic Profile defines the "wire signature" of an operation in a portType to be the fully qualified name of the SOAP Body's child element of the message. For the case of an empty body this name is an empty string. All operations in a port in a DESCRIPTION MUST result in unique wire signatures.*

R2711 *A DESCRIPTION MAY have more than one port with the same value for the* `location` *attribute of the* `soapbind:address` *element.*

When the value of the `location` attribute of two or more `soapbind:address` elements point to the same end-point, and have input messages that are indistinguishable on the wire, implementation problems may arise.

R2712 *When a Doc/literal binding is in use, the wire representation of a MESSAGE MUST be an instance of the global element declaration referenced by that message's single part.*

R2713 *If the value of the* `soapAction` *attribute on the* `soapbind:operation` *element is empty (as indicated by two quotes), the DESCRIPTION MUST be treated equivalent to the one that does not specify the* `soapAction` *attribute.*

R2714 *For one-way operations, INSTANCES MUST NOT return a HTTP response that contains a MESSAGE (i.e., MUST NOT contain a SOAP envelope).*

R2715 *INSTANCES MUST NOT consider one-way operations complete until a HTTP response code of "202 accepted" is received by the HTTP client. In addition the HTTP response code of 202 MUST NOT be interpreted to mean the message is valid or that the receiver would process it.*

R2716 *A DESCRIPTION MUST NOT have the* `namespace` *attribute specified on* `operations` *in SOAP Bindings when* `style`*="document" and* `use`*="literal". This is applicable to all applicable elements of operations; namely* `soap:body`*,* `soap:header`*,* `soap:headerfault` *and* `soap:fault` *elements.*

R2717 *In SOAP Bindings a DESCRIPTION MUST have the* `namespace` *attribute specified on* `operations` *the value of which MUST be an absolute URI, when* `style`*="rpc" and* `use`*="literal". This is applicable to all applicable elements of operations; namely* `soap:body`*,* `soap:header`*,* `soap:headerfault` *and* `soap:fault` *elements.*

R2718 *In a DESCRIPTION the list of operations in portType MUST match that of the corresponding portType in the Binding description.*

R2719 *In a DESCRIPTION specification of `soapbind:headerfault` element in the SOAP Binding description on `wsdl:input` elements or `wsdl:output` elements of an operation is OPTIONAL.*

Eliminate inconsistency between WSDL specification text and the WSDL schema.

R2720 *A DESCRIPTION MUST use the attribute name "part" with a Schema type of "NMTOKEN" for both `soapbind:header` elements and `soapbind:headerfault` elements in the SOAP Binding description on `wsdl:input` element or `wsdl:output` element of an operation.*

The WSDL 1.1 schema is inconsistent with the WSDL 1.1 specification here and incorrectly names the attribute `parts` and gives a type of "NMTOKENS". The schema is incorrect since each `soapbind:header` element references a single part.

CORRECT:

```
<binding name="StockQuoteSoap" type="tns:StockQuotePortType">
  <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="SubscribeToQuotes">
      <input message="tns:SubscribeToQuotes">
        <soap:body parts="body" use="literal"/>
        <soap:header message="tns:SubscribeToQuotes"
                     part="subscribeheader" use="literal"/>
     </input>
   </operation>
</binding>
```

R2721 *A DESCRIPTION MUST have the `name` attribute specified on the `soapbind:fault` element in the SOAP Binding description.*

Eliminate inconsistency between WSDL1.1 specification text and the schema. The WSDL 1.1 schema does not list this attribute.

R2722 *In a DESCRIPTION if the `use` attribute of the `soapbind:fault` element is present in the SOAP Binding description, its value MUST be "literal".*

R2723 *In a DESCRIPTION the specification of the* `use` *attribute on* `soapbind:fault` *elements is OPTIONAL in the SOAP Binding description. If the* `use` *attribute is not present its value MUST be considered equal to "literal".*

To promote interoperability the choice of values for the `use` attribute is limited to "literal".

5.9 XML Schema

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- XML Schema Part 1: Structures.
- XML Schema Part 2: Datatypes.

WSDL/1.1 uses XML Schema as one of its type systems. This profile mandates the use of XML Schema as the type system for WSDL descriptions of Web Services.

R2800 *DESCRIPTIONs MAY use any construct from XML Schema 1.0.*

6. Service Discovery

When discovery is required, UDDI is the mechanism the Basic Profile has adopted to describe Web service providers and the Web services they provide. Business, intended use, and Web service type descriptions are made in UDDI terms; detailed technical descriptions are made in WSDL terms. Where the two specifications define overlapping descriptive data and both forms of description are used, the Basic Profile specifies that the descriptions must not conflict.

UDDI description is optional for Web service instances. By no means do all usage scenarios require the kind of metadata and discovery UDDI provides, but where such capability is needed, UDDI is the sanctioned mechanism.

This portion of the profile incorporates the following specifications by reference;

- The UDDI Version 2.04 API Published Specification, Dated 19 July 2002.
- UDDI Version 2.03 Data Structure Reference, Published Specification, Dated 19 July 2002.
- Version 2.0 UDDI XML Schema 2001.
- UDDI Version 2.03 Replication Specification, Published Specification, Dated 19 July 2002.
- Version 2.03 Replication XML Schema 2001.
- UDDI Version 2.03 XML Custody Schema.

- UDDI Version 2.01Operator's Specification, Published Specification, Dated 19 July 2002.

6.1 businessService Substructure Breakdown

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- UDDI Version 2.03 Data Structure Reference, Section 6.2.

Although UDDI concentrates on business and intended use descriptions and WSDL concentrates on technical descriptions, certain of their data structures are analogous and certain of the data overlap. In particular, the `wsdl:port` element and the `uddi:bindingTemplate` are analogs; both describe a Web service instance and contain the network address of the instance. Similarly, the wsdl:service and the `uddi:businessService` are analogs; they are both sets of instance descriptions.

Since the Basic Profile has adopted both WSDL and UDDI, the profile requires that when both are used the descriptions be parallel and consistent.

> R3000 *The PUBLISHER of a `uddi:businessService` that is also described by a wsdl:service MUST structure the `uddi:businessService` so that every `uddi:bindingTemplate` element is mapped to a wsdl:port and every wsdl:port has a `uddi:bindingTemplate` that maps to it. A `uddi:bindingTemplate` maps to a wsdl:port if and only if the value of its accessPoint attribute is lexically identical to the value of the wsdl:port's location attribute.*

This forces the structure of the `uddi:businessService` and the structure of the wsdl:service to parallel one another. It also forces the network address of the INSTANCEs that are described to be consistent.

> R3001 *The PUBLISHER of a `uddi:businessService` that is also described by a wsdl:service that claims to be a Basic Profile conformant wsdl:service (see requirement Rxxxx) MUST categorize the `uddi:businessService` as being conformant.*

This means that the `categoryBag` element of the businessService must contain a keyedReference using the ws-i-org:conformsTo taxonomy and the categorization of "http://wwww.ws-i.org/profiles/base/1.0"

> **Editors' note:**The above is contingent on the resolution of issue w27 concerning how to mark wsdl elements that conform to the profile. If the value representing the Basic Profile 1.0 changes in the final resolution, the value used here should be changed to match.

6.2 tModel Substructure Breakdown

This portion of the profile modifies and refers to the following specifications (or sections thereof);

- UDDI Version 2.03 Data Structure Reference, Section 8.3.

UDDI represents Web service types as `uddi:tModel` elements. These may, but need not, point (using a URI) to the document that contains the actual description. (See UDDI Data Structures section 8.1.1.) Further, UDDI is agnostic with respect to the mechanisms used to describe Web service types. The Basic Profile cannot be agnostic about this because interoperation is very much complicated if Web service types do not have descriptions or if the descriptions can take arbitrary forms.

The UDDI specification section 13.1.2.1.1 allows but does not require `uddi:tModel` elements that use WSDL to describe the Web service type they represent to state that they use WSDL as the description language. Not doing so leads to interoperability problems because it is then ambiguous what description language is being used.

It is not easy and in some cases it may be impossible to determine whether a given `uddi:tModel` represents a conformant Web service type by inspection alone because `uddi:tModel` elements describing conformant and non-conformant Web service types can look very similar. It needs to be easy for INQUIRERs to determine whether a given `uddi:tModel` conforms and to discover conforming `uddi:tModel` elements.

Therefore the Basic Profile places the following constraints on how `uddi:tModel` elements that describe Web service types may be constructed:

> R3002 *PUBLISHERs of `uddi:tModel` elements representing conformant Web service types MUST use WSDL as the description language.*

This means that a `uddi:tModel` that describes a conformant Web service type must contain an `uddi:overviewDoc` element, the `uddi:overviewDoc` element must contain an `uddi:overviewURL` element, and the `uddi:overviewURL` element must resolve to a conformant WSDL binding describing the Web service type.

> **Editors' note:**For the uddi:overviewURL to resolve to a wsdl:binding, the profile must adopt the convention in the UDDI best practice covering this topic or establish some other convention for distinguishing among multiple wsdl:bindings in a WSDL document. The editor recommends adopting an amended UDDI best practice. See issues u2 and u10.

R3003 *PUBLISHERs of* `uddi:tModel` *elements representing conformant Web service types MUST categorize them to assert that they use WSDL descriptions.*

This means that the `uddi:categoryBag` element of a `uddi:tModel` element representing a conformant Web service type must contain a `uddi:keyedReference` using the uddi-org:types taxonomy and the "wsdlSpec" categorization.

R3004 *PUBLISHERs of* `uddi:tModel`*s MUST construct them so that conformance claims they make are consistent with the conformance claims made by* `wsdl:binding` *elements to which they refer.*

R3005 *PUBLISHERs MUST NOT mark UDDI constructs other than* `uddi:tModel` *elements as being conformant with the Basic Profile.*

This means that the `uddi:categoryBag` element of a `uddi:tModel` representing a conformant Web service type must contain a `uddi:keyedReference` using the ws-i-org:conformsTo taxonomy and the categorization "http://wwww.ws-i.org/profiles/base/1.0" if and only if the WSDL binding to which it refers makes a claim Basic Profile conformance.

Publishers must not mark constructs `uddi:business` and `uddi:service` elements (the only UDDI elements other than `uddi:tModel` that could conceivably be so marked) as being conformant with the Basic Profile because the profile does not define what it means to have a conformant `uddi:business` or a conformant `uddi:businessService`.

Editors' note: The above is contingent on the resolution of issue w27 concerning how to mark wsdl elements that conform to the profile. If the value representing the Basic Profile 1.0 changes in the final resolution, the value used here should be changed to match.

7. Security

As is true of all network-oriented information technologies, the subject of security is a crucial one for Web services. For Web services, as for other information technologies, security consists of understanding the potential threats an attacker may mount and applying operational, physical, and technological countermeasures to reduce the risk of a successful attack to an acceptable level. Because an "acceptable level of risk" varies hugely depending on the application, and because costs of implementing countermeasures is also highly variable, there can be no universal "right answer" for securing Web services. Choosing the absolutely correct balance of countermeasures and acceptable risk can only be done on a case by case basis.

That said, there *are* common patterns of countermeasures that experience shows reduce the risks to acceptable levels for many Web services. The Basic Profile adopts, but does not mandate use of, the most widely used of these: HTTP secured with either TLS 1.0 or SSL

3.0 (HTTPS). That is, conformant Web services may use HTTPS; they may also use other countermeasure technologies or none at all.

HTTPS is widely regarded as mature standard for encrypted transport connections to provide a basic level of confidentiality. HTTPS thus forms the first and simplest means of achieving some basic security features which are required by many real-world web service applications. HTTPS can also be used to provide client authentication through the use of client-side certificates.

This portion of the profile incorporates the following specifications by reference;

- RFC2818: HTTP Over TLS.
- RFC2246: The TLS Protocol Version 1.0.
- The SSL Protocol Version 3.0.
- RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile.

7.1 The Use of HTTPS

R5000 *An INSTANCE MAY require the use of HTTPS.*

R5001 *If an INSTANCE requires the use of HTTPS, the location attribute of the* `soap:address` *element in its* `wsdl:port` *description MUST be a URI whose scheme is "https"; otherwise it MUST be a URI whose scheme is "http".*

R5010 *INSTANCEs MAY require the use of HTTPS with mutual authentication.*

Simple HTTPS provides authentication of the Web service instance by the consumer but not authentication of the consumer by the instance. For many instances this leaves the risk too high to permit interoperation. Including the mutual authentication facility of HTTPS in the profile permits instances to use the countermeasure of authenticating the consumer. In cases in which authentication of the instance by the consumer is insufficient, this often reduces the risk sufficiently to permit interoperation.

7.2 Certificate Authority

R5100 *If an INSTANCE requires use of basic HTTPS, the choice of acceptable certificate authorities for the instance's certificate is a private agreement between the consumer and the instance.*

R5110 *If an INSTANCE requires the use of HTTPS with mutual authentication, the choice of acceptable certificate authorities for the consumer's certificate is a private agreement between the consumer and the instance.*

Successful use of basic HTTPS requires the consumer to agree that the instance's certificate was issued by an acceptable authority. Successful use of mutual authentication

additionally requires the instance to agree that the consumer's certificate was issued by an acceptable authority. The choice of which certificate authorities are acceptable is an important consideration in the effectiveness of using HTTPS, but is a policy decision that is beyond the scope of the profile.

7.3 Permitted HTTPS Encryption Algorithms

R5200 *If an INSTANCE uses HTTPS, the choice of acceptable encryption algorithm is a private agreement between the consumer and the instance.*

Successful use of HTTPS requires the consumer and the instance to agree on a mutually acceptable encryption algorithm. The choice of which encryption algorithms are acceptable is an important consideration in the effectiveness of using HTTPS, but is a policy decision that is beyond the scope of the profile.