Web Services Description Language (WSDL) Version 2.0 Part 3: Bindings

W3C Working Draft 3 August 2004

This version:
http://www.w3.org/TR/2004/WD-wsdl20-bindings-20040803
Latest version:
http://www.w3.org/TR/wsdl20-bindings
Previous versions:
http://www.w3.org/TR/2003/WD-wsdl12-bindings-20030611
Editors:
Hugo Haas, W3C
Philippe Le HÃ©garet, W3C
Jean-Jacques Moreau, Canon
David Orchard, BEA Systems
Jeffrey Schlimmer, Microsoft
Sanjiva Weerawarana, IBM Research

This document is also available in these non-normative formats: postscript, PDF, XML, and plain text.

Copyright Â© 2004 W3CÂ® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Abstract

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. WSDL Version 2.0 Bindings describes how to use WSDL in conjunction with SOAP 1.2 [*SOAP 1.2 Part 1: Messaging Framework [p.37]*] and HTTP/1.1 [*IETF RFC 2616 [p.36]*] (as well as other versions of HTTP). This specification depends on WSDL Version 2.0 [*WSDL 2.0 Core Language [p.37]*].

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is a W3C Last Call Working Draft. If the feedback is positive, the Working Group plans to submit this specification for consideration as a W3C Candidate Recommendation. Comments on this document are invited and are to be sent to the public public-ws-desc-comments@w3.org mailing list (public archive). Comments can be sent until **4 October 2004**.

Three formal objections from Working Group participants have been received against portions of the WSDL 2.0 specification. Feedback is specifically encouraged on these topics:

- Compositors (see objection)
- Feature and properties (see objection and follow-on message)
- Requiring unique GEDs or required feature to distinguish operations (see objection)

A diff-marked version against the previous version of this document is available. For a detailed list of changes since the last publication of this document, please refer to appendix **B. Part 3 Change Log** [p.39] . Issues about this document are documented in the last call issues list maintained by the Working Group.

This document has been produced as part of the W3C Web Services Activity. The authors of this document are the Web Services Description Working Group members.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document has been produced under the 24 January 2002 Current Patent Practice as amended by the W3C Patent Policy Transition Procedure. Patent disclosures relevant to this specification may be found on the Working Group's patent disclosure page. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) with respect to this specification should disclose the information in accordance with section 6 of the W3C Patent Policy.

Short Table of Contents

1. Introduction [p.4]
2. WSDL SOAP Binding [p.7]
3. WSDL HTTP Binding [p.17]
4. References [p.36]
A. Acknowledgements [p.38] (Non-Normative)
B. Part 3 Change Log [p.39] (Non-Normative)

Table of Contents

1. Introduction [p.4]
   1.1 Notational Conventions [p.5]
2. WSDL SOAP Binding [p.7]

2.1 XML Syntax Summary (Non-Normative) [p.8]
2.2 Identifying a SOAP Binding [p.9]
2.3 Default Binding Rules [p.9]
2.4 Specifying the SOAP Underlying Protocol [p.10]
2.4.1 Description [p.11]
2.4.2 Relationship to WSDL Component Model [p.11]
2.4.3 XML Representation [p.11]
2.4.4 Mapping Between Component Properties and XML Representation [p.11]
2.5 Specifying the Default SOAP MEP [p.12]
2.5.1 Description [p.12]
2.5.2 Relationship to WSDL Component Model [p.12]
2.5.3 XML Representation [p.12]
2.6 Declaring SOAP Modules [p.12]
2.6.1 Description [p.12]
2.6.2 Relationship to WSDL Component Model [p.12]
2.6.3 SOAP Module Component [p.13]
2.6.4 XML Representation [p.13]
2.6.5 Mapping Between Component Properties and XML Representation [p.14]
2.7 Binding Faults [p.14]
2.7.1 Description [p.14]
2.7.2 Relationship to WSDL Component Model [p.15]
2.7.3 XML Representation [p.15]
2.7.4 Mapping Between Component Properties and XML Representation [p.15]
2.8 Binding Operations [p.16]
2.8.1 Description [p.16]
2.8.2 Relationship to WSDL Component Model [p.16]
2.8.3 XML Representation [p.16]
2.8.4 Mapping Between Component Properties and XML Representation [p.17]
3. WSDL HTTP Binding [p.17]
3.1 Identifying an HTTP Binding [p.18]
3.2 HTTP Syntax Summary (Non-Normative) [p.18]
3.3 Default Binding Rules [p.19]
3.4 Specifying the HTTP Version [p.20]
3.4.1 Description [p.20]
3.4.2 Relationship to WSDL Component Model [p.20]
3.4.3 XML Representation [p.21]
3.4.4 Mapping Between Component Properties and XML Representation [p.21]
3.5 Specifying the Default HTTP Method [p.21]
3.5.1 Description [p.21]
3.5.2 Relationship to WSDL Component Model [p.21]
3.5.3 XML Representation [p.22]
3.6 Binding Operations [p.22]
3.6.1 Description [p.22]
3.6.2 Relationship to WSDL Component Model [p.22]
3.6.3 XML Representation of HTTP Operation Component [p.23]
3.6.4 Mapping Between HTTP Operation’s XML Representation to Component Properties [p.24]
3.7 Specifying HTTP Error Codes for Faults [p.25]

3.7.1 Description [p.25]
3.7.2 Relationship to WSDL Component Model [p.25]
3.7.3 XML Representation [p.25]
3.7.4 Mapping Between Component Properties and XML Representation [p.26]
3.8 Serialization format of instance data [p.26]
3.8.1 Serialization as application/x-www-form-urlencoded [p.26]
3.8.1.1 Case of elements cited in whttp:location attribute [p.27]
3.8.1.2 Case elements NOT cited in whttp:location attribute [p.27]
3.8.1.2.1 Serialization in the request URI [p.28]
3.8.1.2.2 Serialization in the message body [p.28]
3.8.2 Serialization as application/xml [p.29]
3.8.3 Serialization as multipart/form-data [p.29]
3.9 Operation Styles [p.30]
3.9.1 URI Style [p.30]
3.9.2 Multipart style [p.31]
3.10 Specifying the transfer coding [p.32]
3.10.1 Description [p.32]
3.10.2 Relationship to WSDL Component Model [p.32]
3.10.3 XML Representation [p.32]
3.10.4 Mapping Between Component Properties and XML Representation [p.33]
3.11 Specifying the Use of HTTP Cookies [p.33]
3.11.1 Description [p.33]
3.11.2 Relationship to WSDL Component Model [p.34]
3.11.3 XML Representation [p.34]
3.11.4 Mapping Between Component Properties and XML Representation [p.34]
3.12 Specifying HTTP Access Authentication [p.34]
3.12.1 Description [p.34]
3.12.2 Relationship to WSDL Component Model [p.35]
3.12.3 XML Representation [p.35]
3.12.4 Mapping Between Component Properties and XML Representation [p.36]
4. References [p.36]
4.1 Normative References [p.36]
4.2 Informative References [p.38]

Appendices

A. Acknowledgements [p.38] (Non-Normative)
B. Part 3 Change Log [p.39] (Non-Normative)

---

1. Introduction

The Web Services Description Language WSDL Version 2.0 (WSDL) [*WSDL 2.0 Core Language [p.37]*] defines an XML language for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication. WSDL

2.0 Bindings (this document) defines binding extensions for the following protocols and message formats:

- SOAP Version 1.2 [*SOAP 1.2 Part 1: Messaging Framework [p.37]*] (see **2. WSDL SOAP Binding** [p.7]).
- HTTP/1.1 [*IETF RFC 2616 [p.36]*].

WSDL 2.0 Primer [*WSDL 2.0 Primer [p.38]*] is a non-normative document intended to provide an easily understandable tutorial on the features of the WSDL Version 2.0 specifications.

The Core Language [*WSDL 2.0 Core Language [p.37]*] of the WSDL 2.0 specification describes the core elements of the WSDL language.

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 [*IETF RFC 2119 [p.36]*].

This specification uses a number of namespace prefixes throughout; they are listed in Table 1-1 [p.5] . Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [*XML Information Set [p.37]*]).

Table 1-1. Prefixes and Namespaces used in this specification

Prefix	Namespace	Notes
wsdl	"http://www.w3.org/2004/08/wsdl"	A normative XML Schema [*XML Schema Structures [p.37]*], [*XML Schema Datatypes [p.37]*] document for the "http://www.w3.org/2004/08/wsdl" namespace can be found at http://www.w3.org/2004/08/wsdl.
wsoap	"http://www.w3.org/2004/08/wsdl/soap12"	A normative XML Schema [*XML Schema Structures [p.37]*], [*XML Schema Datatypes [p.37]*] document for the "http://www.w3.org/2004/08/wsdl/soap12" namespace can be found at http://www.w3.org/2004/08/wsdl/soap12.
whttp	"http://www.w3.org/2004/08/wsdl/http"	A normative XML Schema [*XML Schema Structures [p.37]*], [*XML Schema Datatypes [p.37]*] document for the "http://www.w3.org/2004/08/wsdl/http" namespace can be found at http://www.w3.org/2004/08/wsdl/http.
wsdls	"http://www.w3.org/2004/08/wsdl-simple-types"	This prefix and namespace name are used to refer to the simple types defined by [*WSDL 2.0 Core Language [p.37]*] for use in the component model, see Definition of the Simple Types Used in the Component Model.
xs	"http://www.w3.org/2001/XMLSchema"	Defined in the W3C XML Schema specification [*XML Schema Structures [p.37]*], [*XML Schema Datatypes [p.37]*].

Namespace names of the general form "http://example.org/..." and "http://example.com/..." represent application or context-dependent URIs [*IETF RFC 2396 [p.36]*].

All parts of this specification are normative, with the EXCEPTION of pseudo-schemas, examples, and sections explicitly marked as "Non-Normative". Pseudo-schemas are provided for each component, before the description of this component. They provide visual help for the XML 1.0-based [*XML 1.0 [p.37]*] serialization, but do not limit the ability of the bindings to be expressed with other versions of XML such as XML 1.1 [*XML 1.1 [p.38]*] or futures versions of XML Schema [*XML Schema Structures [p.37]*], [*XML Schema Datatypes [p.37]*].

Note that the WSDL binding specifications in this document are defined in terms of a component model defined by this specification. As such, it is explicitly NOT a conformance requirement to be able to process documents encoded in a particular version of XML.

2. WSDL SOAP Binding

The SOAP binding described in this section is an extension for [*WSDL 2.0 Core Language [p.37]*] to enable Web Services applications to use SOAP 1.2 [*SOAP 1.2 Part 1: Messaging Framework [p.37]*]. This binding extends WSDL 2.0 by adding properties to the Binding component as defined in [*WSDL 2.0 Core Language [p.37]*]. In addition, an XML Infoset representation for these additional properties is provided, along with a mapping from that representation to the various component properties.

As allowed in [*WSDL 2.0 Core Language [p.37]*], a Binding component MAY exist without indicating a specific Interface component that it applies to. In this case there MUST NOT be any Binding Operation or Binding Fault components present in the Binding component.

The SOAP binding is designed with the objective of minimizing what needs to be explicitly declared for common cases. This is achieved by defining a set of default rules which apply for all Interface Operation components of an Interface component, unless specifically overidden on a per Interface Operation basis. Thus, if a given Interface Operation component is not referred to specifically, then all the default rules apply for that component. That is, per the requirements of [*WSDL 2.0 Core Language [p.37]*] all operations of an Interface component are bound by this binding.

Notice that there are no default binding rules defined for Interface Fault components by this binding. Thus, if a given Interface component has any Fault components, then such Interface components MUST be bound via Binding components which indicate a specific interface and contain as many Binding Fault components as there are Fault components in the Interface Fault component.

A subset of the HTTP properties specified in the HTTP binding defined in section **3. WSDL HTTP Binding** [p.17] may be expressed in a SOAP binding when the SOAP binding uses HTTP as the underlying protocol, i.e. when the value of the {soap underlying protocol} property of the Binding component is "http://www.w3.org/2003/05/soap/bindings/HTTP/". The properties that are allowed are the ones that describe the underlying protocol.

- {http version} as defined in **3.4 Specifying the HTTP Version** [p.20]
- {http transfer coding} as defined in **3.10 Specifying the transfer coding** [p.32]
- {http cookies} as defined in **3.11 Specifying the Use of HTTP Cookies** [p.33]
- {http authentication scheme} and {http authentication realm} as defined in **3.12 Specifying HTTP Access Authentication** [p.34]

When the SOAP Message Exchange Pattern is the SOAP Response MEP, the Binding Operation may use the {http location} property defined in **3.6 Binding Operations** [p.22] . When such a location is specified, the Endpoint component also follows the rules for constructing the address from the {address} property and the {http location} property values.

2.1 XML Syntax Summary (Non-Normative)

```
<definitions >
  <binding name="xs:NCName" interface="xs:QName"?
           type="http://www.w3.org/2004/08/wsdl/soap12"
           whttp:version="xs:string"??
           whttp:defaultTransferCoding="xs:string"??
           wsoap:protocol="xs:anyURI"
           wsoap:mepDefault="xs:anyURI"? >
    <documentation />?

    <wsoap:module uri="xs:anyURI" required="xs:boolean"? >
      <documentation />?
    </wsoap:module>*

    <fault ref="xs:QName"
           wsoap:code="xs:QName"
           wsoap:subcodes="list of xs:QName"? >
      <documentation />?
    </fault>*

    <operation ref="xs:QName"
               whttp:defaultTransferCoding="xs:string"?? >
               wsoap:mep="xs:anyURI"?
               wsoap:action="xs:anyURI"? >
      <documentation />?

      <wsoap:module ... />*

      <input messageLabel="xs:NCName"?
             whttp:transferCoding="xs:string?? >
        <documentation />?
        <wsoap:module ... />*
        <feature ... />*
        <property ... />*
      </input>*

      <output messageLabel="xs:NCName"?
             whttp:transferCoding="xs:string?? >
        <documentation />?
        <wsoap:module ... />*
        <feature ... />*
        <property ... />*
      </output>*

      <feature ... />*
      <property ... />*
    </operation>*

    <feature ... />*
    <property ... />*

  </binding>

  <service>
    <endpoint name="xs:NCName" binding="xs:QName" address="xs:anyURI"?
              whttp:authenticationType="xs:string"??
```

```
            whttp:authenticationRealm="xs:string"?? >
      <documentation />?
      <feature ... />*
      <property ... />*
    </endpoint>
    <feature ... />*
    <property ... />*
  </service>
</definitions>
```

Note:

The double question marks ("??") after the attributes in the `whttp` namespace indicates that those optional attributes only make sense if the when the SOAP binding uses HTTP as the underlying protocol, i.e. when the value of the `wsoap:protocol` attribute is "http://www.w3.org/2003/05/soap/bindings/HTTP/".

2.2 Identifying a SOAP Binding

A Binding component (defined in [*WSDL 2.0 Core Language [p.37]*]) is identified as a SOAP binding by assigning the value "http://www.w3.org/2004/08/wsdl/soap12" to the {type} property of the Binding component.

2.3 Default Binding Rules

- *Payload Construction.* When formulating the SOAP envelope to be transmitted the contents of the payload (i.e., the contents of the `soap:Body` *element information item* of the SOAP envelope) MUST be what is defined by the corresponding Message Reference component. This is subject to optimization by a feature that is in use which may affect serialization, such as MTOM [*SOAP Message Transmission Optimization Mechanism [p.38]*]. The following default binding rules MUST be adhered to:

  - If the value of the {message content model} property of the Message Reference component is *#any* then the payload MAY be any one XML element.

  - If the value is *#none* then the payload MUST be empty.

  - If the value is *#element* then the payload will be the *element information item* identified by the {element} property of the Message Reference component.

  - If the Message Reference component is declared using a non-XML type system (as considered in the Types section of [*WSDL 2.0 Core Language [p.37]*]) then additional binding rules MUST be defined to indicate how to map those components into the SOAP envelope.

  Note:

  This SOAP binding only allows one single element in SOAP body.

- *SOAPAction.* If a value for the {soap action} property of a Binding Operation component has NOT been specified then the SOAP Action Feature (see [*SOAP 1.2 Part 2: Adjuncts [p.37]*]) has NO value assigned by the Binding component.

- *SOAP MEP Selection.* If the Interface Operation component's {message exchange pattern} property has the value "http://www.w3.org/2004/08/wsdl/in-out" then the default value of the {soap mep} property for the corresponding Binding Operation component is "http://www.w3.org/2003/05/soap/mep/request-response/" identifying the SOAP Request-Response Message Exchange Pattern as defined in [*SOAP 1.2 Part 2: Adjuncts [p.37]*]. If the Inteface Operation component has any other value for the {message exchange pattern} property, then no default value is defined for the {soap mep} property of the corresponding Binding Operation component.

- *HTTP Method Selection.* This default binding rule is applicable when the value of the {soap underlying protocol} property of the Binding component is "http://www.w3.org/2003/05/soap/bindings/HTTP/". If the {soap mep} property of the Binding Operation component has the value "http://www.w3.org/2003/05/soap/mep/request-response/" then the default value of the {http method} property is *POST*. If the {soap mep} property of the Binding Operation component has the value "http://www.w3.org/2003/05/soap/mep/soap-response/" then the default value of the {http method} property is *GET*.

- *HTTP URI Generation.* This default binding rule is applicable when the value of the {soap underlying protocol} property of the Binding component is "http://www.w3.org/2003/05/soap/bindings/HTTP/". If the {soap mep} property of the Binding Operation component has the value "http://www.w3.org/2003/05/soap/mep/soap-response/" then the URI to execute the HTTP GET against MUST be generated using the HTTP binding's rules for generating a URI for HTTP GET (see **3. WSDL HTTP Binding** [p.17]). The input serialization format of `x-www-form-urlencoded` is the only supported serialization format for HTTP GET in the SOAP Response Message Exchange Pattern.

Editorial note: Input serialization for HTTP GET in SOAP HTTP binding	
Use of a different input serialization format requires introduction of either a new MEP or a new binding. The Working Group considered the limitations of the `x-www-form-urlencoded` serialization format (see points #2 and #3 of Binding message content to URI analysis). It decided that the limitations of the serialization format, which could potentially be solved by a serialization format extension, were not sufficiently broad enough to warrant allowing extensibility in input serialization for the soap-response MEP. The Working Group solicits the public's feedback on this decision.	

2.4 Specifying the SOAP Underlying Protocol

2.4.1 Description

Every SOAP binding MUST indicate what underlying protocol is in use, as defined in the SOAP Protocol Binding Framework section of [*SOAP 1.2 Part 1: Messaging Framework [p.37]*].

The SOAP binding defined by this specification supports the SOAP HTTP binding defined by the [*SOAP 1.2 Part 2: Adjuncts [p.37]*] specification. This is indicated by assigning the URI "http://www.w3.org/2003/05/soap/bindings/HTTP/" (as defined by [*SOAP 1.2 Part 2: Adjuncts [p.37]*]) to the {soap underlying protocol} property defined below. Other values MAY be used for this property in conjunction with the SOAP binding defined by this specification provided that the semantics of such protocols are consistent with this binding.

2.4.2 Relationship to WSDL Component Model

The SOAP protocol specification adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {soap underlying protocol}, an absolute URI as defined by [*IETF RFC 2396 [p.36]*], to the Binding component.

2.4.3 XML Representation

```
<definitions >
  <binding  name="xs:NCName" interface="xs:QName"? type="xs:anyURI"
            wsoap:protocol="xs:anyURI" >
    ...
  </binding>
</definitions>
```

The XML representation for specifying the SOAP protocol is a REQUIRED *attribute information item* with the following Infoset properties:

- A [local name] of `protocol`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/soap12"

2.4.4 Mapping Between Component Properties and XML Representation

See Table 2-1 [p.11] .

Table 2-1. Mapping between Binding Component Extension Properties and XML Representation

Property	Mapping
{soap underlying protocol}	The actual value of the `wsoap:protocol` *attribute information item.*

2.5 Specifying the Default SOAP MEP

2.5.1 Description

Every Binding Operation component of a SOAP binding MUST indicate the SOAP Message Exchange Pattern (MEP) (see [*SOAP 1.2 Part 2: Adjuncts [p.37]*]) to be used for that operation. This binding specification allows the user to indicate a default SOAP MEP to be used for all Binding Operation components of this Binding component.

2.5.2 Relationship to WSDL Component Model

The default SOAP MEP specification is a syntactic convenience and does not affect the underlying component model.

2.5.3 XML Representation

```
<definitions >
  <binding  name="xs:NCName" interface="xs:QName"? type="xs:anyURI"
            wsoap:protocol="xs:anyURI"
            wsoap:mepDefault="xs:anyURI ?" >
    ...
  </binding>
</definitions>
```

The XML representation for specifying the default SOAP MEP is an OPTIONAL *attribute information item* with the following Infoset properties:

- A [local name] of `mepDefault`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/soap12"

2.6 Declaring SOAP Modules

2.6.1 Description

In SOAP, it is permissible for specification interaction to engage one or more additional features (typically implemented as one or more SOAP header blocks), as defined by SOAP Modules (see [*SOAP 1.2 Part 1: Messaging Framework [p.37]*]). This binding specification allows users to indicate which SOAP Modules are in use across the entire binding, on a per operation basis or on a per message basis.

2.6.2 Relationship to WSDL Component Model

The SOAP Module component adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {soap modules} to the Binding, Binding Operation and Binding Message Reference components.

The SOAP modules applicable for a particular operation of any service consists of all modules specified in the input or output Binding Message reference components, and those specified within the Binding Operation components and those specified within the Binding component. If any module is declared in multiple components, then the requiredness of that module is defined by the closest declaration, where closeness is defined by whether it is specified directly at the Binding Message Reference component level, the Binding Operation component level or the Binding component level, respectively.

2.6.3 SOAP Module Component

The SOAP Module component identifies a SOAP module that is in use. The properties of the SOAP Module component are as follows:

- {uri} An absolute URI as defined by [*IETF RFC 2396 [p.36]*]. The value of this property identifies the specific SOAP module that is in use, as defined in the SOAP Protocol Binding Framework section of [*SOAP 1.2 Part 1: Messaging Framework [p.37]*].
- {required} A boolean indicating if the SOAP module is required.

2.6.4 XML Representation

```
<definitions >
  <binding >
    <wsoap:module uri="uri"
                  required="boolean"? >
      <documentation ... />?
    </wsoap:module>
    <operation>
      <wsoap:module ... />*
      <input>
        <wsoap:module ... />*
      </input>
      <output>
        <wsoap:module ... />*
      </output>
    </operation>
  </binding>
</definitions>
```

The XML representation for a SOAP Module component is an *element information item* with the following Infoset properties:

- A [local name] of `module`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/soap12"
- One or more *attribute information item*s amongst its [attributes] as follows:
    - A REQUIRED `uri` *attribute information item* with the following Infoset properties:

    - A [local name] of `uri`
    - A [namespace name] which has no value
  - An OPTIONAL `required` *attribute information item* with the following Infoset properties:
    - A [local name] of `required`
    - A [namespace name] which has no value
  - Zero or more namespace qualified *attribute information item*s. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/08/wsdl" and MUST NOT be "http://www.w3.org/2004/08/wsdl/soap12".
- Zero or more *element information item* amongst its [children], in order, as follows:
  1. An OPTIONAL `documentation` *element information item* as defined in [*WSDL 2.0 Core Language [p.37]*].
  2. Zero or more namespace-qualified *element information item*s amongst its [children]. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2004/08/wsdl" and MUST NOT be "http://www.w3.org/2004/08/wsdl/soap12".

2.6.5 Mapping Between Component Properties and XML Representation

See Table 2-2 [p.14] .

Table 2-2. Mapping between SOAP Module Component Properties and XML Representation

Property	Mapping
{uri}	The actual value of the `uri` *attribute information item*.
{required}	The actual value of the `required` *attribute information item*.

2.7 Binding Faults

2.7.1 Description

For every Interface Fault component contained in an Interface component, a mapping to a SOAP Fault must be described. This binding specification allows the user to indicate the SOAP fault code and subcodes that are transmitted for a given Interface Fault component.

2.7.2 Relationship to WSDL Component Model

The SOAP Fault binding adds the following properties to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {soap fault code}, a QName as defined by [*XML 1.0 [p.37]*], to the Binding Fault component. The value of this property identifies a possible SOAP fault (see [*SOAP 1.2 Part 1: Messaging Framework [p.37]*]) for the operation in scope.
- {soap fault subcodes}, a list of QNames, to the Binding Fault component. The value of this property identifies one or more subcodes for this SOAP fault.

2.7.3 XML Representation

```
<definitions >
  <binding >
    <fault ref="xs:QName"
           wsoap:code="xs:QName"
           wsoap:subcodes="list of xs:QName"? />
    </fault>*
  </binding>
</definitions>
```

The XML representation for binding a SOAP Fault are two *attribute information item*s with the following Infoset properties:

- wsoap:code REQUIRED *attribute information item*
  - A [local name] of `code`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/soap12"
- wsoap:subcodes OPTIONAL *attribute information item*
  - A [local name] of `subcodes`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/soap12"

2.7.4 Mapping Between Component Properties and XML Representation

See Table 2-3 [p.15] .

Table 2-3. Mapping between SOAP Fault Component Properties and XML Representation

Property	Mapping
{soap fault code}	The actual value of the `code` *attribute information item*.
{soap fault subcodes}	The actual value of the `subcodes` *attribute information item*.

2.8 Binding Operations

2.8.1 Description

For every Interface Operation component contained in an Interface component, in addition to the default binding rules described in **2.3 Default Binding Rules** [p.9] , there may be additional binding information to be specified. This binding specification allows the user to indicate the SOAP Message Exchange Pattern (MEP) and a value for the SOAP Action Feature (see [*SOAP 1.2 Part 2: Adjuncts [p.37]*]) on a per-operation basis.

2.8.2 Relationship to WSDL Component Model

The SOAP Operation binding specification adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {soap mep}, an absolute URI as defined by [*IETF RFC 2396 [p.36]*], to the Binding Operation component. The value of this property identifies the SOAP Message Exchange Pattern (MEP) (as defined by [*SOAP 1.2 Part 1: Messaging Framework [p.37]*]) for this specific operation. If no specific value is assigned, then the value assigned by the default rules apply (see **2.3 Default Binding Rules** [p.9]). It is an error for this property to not have a value (which MAY happen if the default rules are not applicable).

- {soap action}, an absolute URI as defined by [*IETF RFC 2396 [p.36]*], to the Binding Operation component. The value of this property identifies the value of the SOAP Action Feature (as defined by [*SOAP 1.2 Part 1: Messaging Framework [p.37]*]) for this specific operation.

2.8.3 XML Representation

```
<definitions >
  <binding >
    <operation ref="xs:QName"
               wsoap:mep="xs:anyURI"?
               wsoap:action="xs:anyURI"? >
    </operation>
  </binding>
</definitions>
```

The XML representation for binding an Operation are two *attribute information item*s with the following Infoset properties:

- wsoap:mep OPTIONAL *attribute information item*
  - A [local name] of `mep`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/soap12"
- wsoap:action OPTIONAL *attribute information item*

- A [local name] of `action`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/soap12"

2.8.4 Mapping Between Component Properties and XML Representation

See Table 2-4 [p.17] .

Table 2-4. Mapping between SOAP Operation Component Properties and XML Representation

Property	Mapping
{soap mep}	The actual value of the `wsoap:mep` *attribute information item*, if present. If not, the actual value of the `wsoap:mepDefault` *attribute information item* of the parent wsdl:binding *element information item*, if present. If not the value as defined by the default SOAP binding rules (see **2.3 Default Binding Rules** [p.9]), if applicable.
{soap action}	The actual value of the `action` *attribute information item*., if any.

3. WSDL HTTP Binding

The HTTP binding described in this section is an extension for [*WSDL 2.0 Core Language [p.37]*] to enable Web Services applications to use HTTP 1.1 [*IETF RFC 2616 [p.36]*] (as well as other versions of HTTP) and HTTPS [*IETF RFC 2818 [p.36]*]. This binding extends WSDL 2.0 by adding properties to the component model defined in [*WSDL 2.0 Core Language [p.37]*]. In addition an XML Infoset representation for these additional properties is provided, along with a mapping from that representation to the various component properties.

As allowed in [*WSDL 2.0 Core Language [p.37]*], a Binding component MAY exist without indicating a specific Interface component that it applies to. In this case there MUST NOT be any Binding Operation or Binding Fault components present in the Binding component.

The HTTP binding is designed with the objective of minimizing what needs to be explicitly declared for common cases. This is achieved by defining a set of default rules which apply for all Interface Operation components of an Interface component, unless specifically overidden on a per Interface Operation basis. Thus, if a given Interface Operation component is not referred to specifically, then all the default rules apply for that component. That is, per the requirements of [*WSDL 2.0 Core Language [p.37]*] all operations of an Interface component are bound by this binding.

Notice that there are no default binding rules defined for Fault components by this binding. Thus, if a given Interface component has any Fault components, then such Interface components MUST be bound via Binding components which indicate a specific interface and contain as many Binding Fault components as there are Fault components in the Interface Fault component.

[Definition: The internal tree representation of an input, output or fault message is called an **instance data**, and is constrained by the schema definition associated the message: the XML element referenced in the {element} property of the Message Reference component for input and output messages, and in the {element} property of an Interface Fault component for faults.]

3.1 Identifying an HTTP Binding

A Binding component (defined in [*WSDL 2.0 Core Language [p.37]*]) is identified as an HTTP binding by assigning the value "http://www.w3.org/2004/08/wsdl/http" to the {type} property of the Binding component.

3.2 HTTP Syntax Summary (Non-Normative)

```
<definitions >
  <binding name="xs:NCName" interface="xs:QName"? type="xs:anyURI"
           whttp:methodDefault="xs:string"?
           whttp:cookies="xs:boolean"?
           whttp:version="xs:string"?
           whttp:defaultTransferCoding="xs:string"? >
   <documentation />?

    <fault ref="xs:QName"
           whttp:code="xs:int"? >
      <documentation />?
    </fault>*

    <operation ref="xs:QName"
               whttp:location="xs:anyURI"?
               whttp:method="xs:string"?
               whttp:inputSerialization="xs:string"?
               whttp:outputSerialization="xs:string"?
               whttp:faultSerialization="xs:string"?
               whttp:defaultTransferCoding="xs:string"? >
          <documentation />?

      <input messageLabel="xs:NCName"?
             whttp:transferCoding="xs:string? >
        <documentation />?>*
        <feature ... />*
        <property ... />*
      </input>*

      <output messageLabel="xs:NCName"?
              whttp:transferCoding="xs:string? >
        <documentation />?
        <feature ... />*
        <property ... />*
      </output>*
    </operation>*

  </binding>

  <service>
    <endpoint name="xs:NCName" binding="xs:QName" address="xs:anyURI"?
```

```
            whttp:authenticationType="xs:string"?
            whttp:authenticationRealm="xs:string"? >
      <documentation />?
    </endpoint>
  </service>
</definitions>
```

3.3 Default Binding Rules

- *HTTP Method Declaration.* When formulating the HTTP message to be transmitted, the HTTP request method MUST be what is defined by the `whttp:method` attribute on `operation` , or with the `whttp:defaultMethod` attribute on `binding` .

- *Payload construction.* When formulating the HTTP message to be transmitted, the contents of the payload (i.e. the contents of the HTTP message body) MUST be what is defined by the corresponding Message Reference or Interface Fault components:

  - Message Reference component: if the value of the {message content model} property is *#any* then the payload MAY be any one XML element. If the value is *#none* then the payload MUST be empty. Finally if the value is *#element* then the payload will be the *element information item* identified by the {element} property.

  - Interface Fault component: the payload will be the *element information item* identified by the {element} property.

  If the Message Reference component or the Interface Fault component is declared using a non-XML type system (as considered in the Types section of [*WSDL 2.0 Core Language [p.37]*]) then additional binding rules MUST be defined to indicate how to map those components into the HTTP envelope.

- *Serialization format.* The HTTP request serialization format MUST be what is defined by the {http input serialization} property. The HTTP response serialization format MUST be what is defined by the {http output serialization} property. The HTTP serialization format of a fault MUST be what is defined by the {http fault serialization} property.

  Section **3.8 Serialization format of instance data** [p.26] defines serialization formats supported by this binding along with their constraints.

- *Default input and output serialization format.* Table 3-1 [p.19] defines the default values for the GET, POST, PUT and DELETE values of the {http method} property.

Table 3-1. Default values for GET, POST, PUT and DELETE

HTTP Method	Default Input Serialization	Default Output Serialization
{http method}	**{http input serialization}**	**{http output serialization}**
GET	`application/x-www-form-urlencoded`	`application/xml`
POST	`application/xml`	`application/xml`
PUT	`application/xml`	`application/xml`
DELETE	`application/x-www-form-urlencoded`	`application/xml`

Note:

The `application/x-www-form-urlencoded` serialization format places constraints on the stype of the interface operation bound (see **3.8.1 Serialization as application/x-www-form-urlencoded** [p.26]).

The default vales for the {http input serialization} and {http output serialization} properties for any other {http method} is `application/xml`.

Mechanisms that are outside the scope of this specification MAY modify the serialization format of the instance data [p.18] corresponding to the output message. An example of such modification is the combination of the serialization as `application/x-www-form-urlencoded` and the SOAP-Response Message Exchange Pattern ([*SOAP 1.2 Part 2: Adjuncts [p.37]*], Section 6.3).

- *Accept headers.* Standard HTTP accept headers (see section 14 of [*IETF RFC 2616 [p.36]*]) MAY be used in an HTTP request. When constructing an HTTP `Accept` header, the requester agent MAY take into account the `expectedMediaType` information (see [*MTXML [p.38]*]) appearing on an output message description to find out about the type of binary element content which is expected to be sent by the provider agent.

3.4 Specifying the HTTP Version

3.4.1 Description

Every Binding component MUST indicate what version of HTTP is in use for the operations of the interface that this binding applies to.

By default, HTTP/1.1 [*IETF RFC 2616 [p.36]*] is used.

3.4.2 Relationship to WSDL Component Model

The HTTP binding specification adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {http version}, a string value to the Binding component.

3.4.3 XML Representation

```
<definitions >
  <binding name="xs:NCName" interface="xs:QName"? type="xs:anyURI"
           whttp:version="xs:string"? >
  </binding>
</definitions>
```

The XML representation for specifying the HTTP version is an optional *attribute information item* with the following Infoset properties:

- A [local name] of `version`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
- A type of *wsdls:string*
- A default value of *"1.1"*

3.4.4 Mapping Between Component Properties and XML Representation

See Table 3-2 [p.21] .

Table 3-2. Mapping between Binding Component Extension Properties and XML Representation

Property	Mapping
{http version}	The actual value of the `whttp:version` *attribute information item*.

3.5 Specifying the Default HTTP Method

3.5.1 Description

Every Binding Operation component MUST indicate what HTTP method is in use for the operations of the interface that this binding applies to. This binding specification allows the user to indicate a default HTTP method to be used for all Binding Operation components of this Binding component.

3.5.2 Relationship to WSDL Component Model

The default HTTP method specification is a syntactic convenience and does not affect the underlying component model.

3.5.3 XML Representation

```
<definitions >
  <binding name="xs:NCName" interface="xs:QName"? type="xs:anyURI"
           whttp:defaultMethod="xs:string"? >
  </binding>
</definitions>
```

The XML representation for specifying the default HTTP method is an optional *attribute information item* with the following Infoset properties:

- A [local name] of `defaultMethod`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
- A type of *wsdls:string*
- No default value

3.6 Binding Operations

3.6.1 Description

This binding specification provides a binding to HTTP of Interface Operation components whose {message exchange pattern} property has the value ’http://www.w3.org/2004/08/wsdl/in-only’, ’http://www.w3.org/2004/08/wsdl/robust-in-only’ or ’http://www.w3.org/2004/08/wsdl/in-out’.

For every Binding operation component corresponding to such Interface Operation components, this binding specification allows the user to indicate the HTTP method to use, the input, output and fault serialization, and the location of the bound operation.

3.6.2 Relationship to WSDL Component Model

The HTTP Operation component adds the following property to the Binding Operation component of the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {http location}, an absolute or relative URI as defined by [*IETF RFC 2396 [p.36]*]. The value of this property specifies a template for the relative portion of the request URI for an operation. This URI is combined with the base URI specified in the {address} property of the endpoint element to form the full URI for the HTTP request to invoke the operation. It MUST contain an absolute or a relative URI, i.e. it MUST NOT include a fragment identifier in the URI.

  If the resulting URI uses the `https` scheme, then HTTP over TLS [*IETF RFC 2818 [p.36]*] is used to send the HTTP request.

- {http method}, a string value indicating, if present, the value for the HTTP Request Method for this specific operation. Otherwise, the default HTTP method as defined in **3.5 Specifying the Default HTTP Method** [p.21] applies. One or the other MUST be present

- {http input serialization}, a string value indicating, if present, the value for the serialization of the HTTP Request message for this specific operation. Its value MUST be the name of a IANA media type token. If not present, the default input serialization associated with the {http method} property applies, as specified in **3.3 Default Binding Rules** [p.19] .
- {http output serialization}, a string value indicating, if present, the value for the serialization of the HTTP Response message for this specific operation. Its value MUST be the name of a IANA media type token. If not present, the default output serialization associated with the {http method} property applies, as specified in **3.3 Default Binding Rules** [p.19] .
- {http fault serialization}, a string value indicating the value for the serialization of the HTTP Response message for this specific operation in case a fault is returned. Its value MUST be the name of a IANA media type token.

3.6.3 XML Representation of HTTP Operation Component

```
<definitions>
 <binding>
   <operation ref="xs:QName"
              whttp:location="xs:anyURI"?
              whttp:method="xs:string"?
              whttp:inputSerialization="xs:string"?
              whttp:outputSerialization="xs:string"?
              whttp:faultSerialization="xs:string"? >
  </operation>
 </binding>
</definitions>
```

The XML representation for binding an Operation are four *attribute information item*s with the following Infoset properties:

- An OPTIONAL `location` *attribute information item* with the following Infoset properties:
  - A [local name] of `location`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
  - A type of *wsdls:anyURI*
  - No default value
- An OPTIONAL `method` *attribute information item* with the following Infoset properties:
  - A [local name] of `method`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
  - A type of *wsdls:string*

  - No default value

- An OPTIONAL `inputSerialization` *attribute information item* with the following Infoset properties:
  - A [local name] of `inputSerialization`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
  - A type of *wsdls:string*
  - No default value
- An OPTIONAL `outputSerialization` *attribute information item* with the following Infoset properties:
  - A [local name] of `outputSerialization`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
  - A type of *wsdls:string*
  - No default value
- An OPTIONAL `faultSerialization` *attribute information item* with the following Infoset properties:
  - A [local name] of `faultSerialization`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
  - A type of *wsdls:string*
  - A default value of *"application/xml"*

3.6.4 Mapping Between HTTP Operation's XML Representation to Component Properties

See Table 3-3 [p.24] .

Table 3-3. Mapping between Binding Operation Component Extension Properties and XML Representation

Property	Mapping
{http location}	The actual value of the `whttp:location` *attribute information item*.
{http method}	The actual value of the `whttp:method` *attribute information item*.
{http input serialization}	The actual value of the `whttp:inputSerialization` *attribute information item*.
{http output serialization}	The actual value of the `whttp:outputSerialization` *attribute information item*.
{http fault serialization}	The actual value of the `whttp:faultSerialization` *attribute information item*.

3.7 Specifying HTTP Error Codes for Faults

3.7.1 Description

For every Interface Fault component contained in an Interface component, an HTTP error code MAY be defined. It represents the error code that will be used by the service in case the fault needs to be returned.

3.7.2 Relationship to WSDL Component Model

The HTTP Fault binding adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {http error status code}, an integer representing a error Status-Code as defined by [*IETF RFC 2616 [p.36]*], to the Binding Fault component. The value of this property identifies the error code that the service will use in case the fault is returned. If empty, no claim is made by the service.

3.7.3 XML Representation

```
<definitions >
  <binding >
    <fault ref="xs:QName"
           whttp:code="xs:int" />
    </fault>*
  </binding>
</definitions>
```

The XML representation for binding a SOAP Fault are two *attribute information item*s with the following Infoset properties:

- whttp:code OPTIONAL *attribute information item*

- A [local name] of `code`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
- A type of *wsdls:int*

3.7.4 Mapping Between Component Properties and XML Representation

See Table 3-4 [p.26] .

Table 3-4. Mapping between Binding Fault Component Extension Properties and XML Representation

Property	Mapping
{http error status code}	The actual value of the `whttp:code` *attribute information item*.

3.8 Serialization format of instance data

The following serialization formats can be used to encode the instance data [p.18] corresponding to the input and output message, as well as the media types and HTTP headers associated.

Other serialization formats may be used. Those MAY place restrictions on the style of the interface operation bound.

3.8.1 Serialization as "application/x-www-form-urlencoded"

This serialization format is designed to allow a Web Service to produce a URI based on the instance data [p.18] of input messages. It may only be used for interface operation using the URI Style format as defined in **3.9.1 URI Style** [p.30] .

Elements from the instance data can be inserted into the path of the request URI, or a query parameter, as shown in the example below:

Example 3-1. Instance data serialized in a URI

The following instance data of an input message

```
<data>
  <town>FrÃ©jus</town>
  <date>2004-01-16</date>
  <unit>C</unit>
</data>
```

with the following `operation` element

```
<operation ref='t:io'
    whttp:location='temperature/{town}'
    whttp:method='GET' />
```

and the following `endpoint` element

```
<endpoint name='e' binding='t:b'
    address='http://ws.example.com/service1' />
```

will serialize the message in the URI as follow:

```
http://ws.example.com/service1/temperature/Fr%C3%A9jus?date=2004-01-16&unit=C
```

Note:

Element name and element content MUST be URI escaped when inserted into the request, as defined in "Serialization as application/x-www-form-urlencoded" ([*XForms 1.0 [p.37]*], Section 11.6).

3.8.1.1 Case of elements cited in `whttp:location` attribute

Editorial note: URIPath Feedback Requested	
The inclusion of elements of the instance data in the path of the request URI, whilst supported by WSDL 1.1, is not supported by XForms 1.0. Hence this mechanism MAY be removed in a future version of this specification. Feedback on this issue from users and implementers is highly encouraged.	

The `location` *attribute information item* MAY cite elements instance data [p.18] of the input message to be serialized in the path of the request URI ("URI Syntactic Components", [*IETF RFC 2396 [p.36]*], Section 3) by enclosing the element name within curly braces (e.g. location="temperature/{town}"):

- When constructing the request URI, each pair of curly braces (and enclosed element name) is replaced by the corresponding content of the element.
- A double curly brace (i.e. "{{" or "}}") MAY be used to include a single, literal curly brace in the request URI.

An element MUST NOT be cited more than once within the `location` *attribute information item*.

An element name MAY be followed by a slash (i.e. "/") inside curly braces (e.g. location="temperature/{town/}") to indicate that no other element must be serialized in the request URI (see **3.8.1.2 Case elements NOT cited in whttp:location attribute** [p.27]).

Strings enclosed within single curly braces MUST be element names from the instance data [p.18] of the input message, possibly followed by a slash; any other strings enclosed within single curly braces are a fatal error.

3.8.1.2 Case elements NOT cited in `whttp:location` attribute

If not all elements from the instance data [p.18] are cited in the `whttp:location` attribute, then additional serialization rules apply.

If an element name appears in the `whttp:location` *attribute information item* followed by a slash, then the instance data must be serialized in the message body (see **3.8.1.2.2 Serialization in the message body** [p.28]), otherwise the elements not cited must be serialized as parameters in the request URI (see **3.8.1.2.1 Serialization in the request URI** [p.28]).

3.8.1.2.1 Serialization in the request URI

All elements of the instance data [p.18] from the input message NOT cited by the `location` *attribute information item* are serialized as query parameters appended to the request URI (e.g. Example 3-1 [p.26]).

If the `location` attribute does not contain a '?' (question mark) character, one is appended. If it does already contain a question mark character, then an "&" separator character is appended. Each parameter pair is separated by the "&" separator character.

- Uncited elements with single values (non-list) are serialized as a single name-value parameter pair. The name of the parameter is the name of the uncited element, and the value of the parameter is the value of the uncited element.
- Uncited elements with list values are serialized as one name-value parameter pair per list value. The name of each parameter is the name of the uncited element, and the value of each parameter is the corresponding value in the list.

3.8.1.2.2 Serialization in the message body

In addition to the serialization in the request URI of the elements cited in the `whttp:location` attribute, the entire instance data [p.18] is serialized in the message body following the rules of the "application/xml" (see **3.8.2 Serialization as application/xml** [p.29]).

Example 3-2. Instance data serialized in a URI and in a message body

The following instance data of an input message

```
<data>
  <town>FrÃ©jus</town>
  <date>2004-01-16</date>
  <unit>C</unit>
  <value>24</value>
</data>
```

with the following `operation` element:

```
<operation ref='t:io'
    whttp:inputSerialization='application/x-www-form-urlencoded'
    whttp:location='temperature/{town/}'
    whttp:method='POST' />
```

and the following `endpoint` element

```
<endpoint name='e' binding='t:b'
    address='http://ws.example.com/service1' />
```

will serialize the message in the URI as follow:

```
http://ws.example.com/service1/temperature/Fr%C3%A9jus
```

and in the message as follow:

```
Content-Type: application/xml
Content-Length: xxx

<data>
  <town>FrÃ©jus</town>
  <date>2004-01-16</date>
  <unit>C</unit>
  <value>24</value>
</data>
```

3.8.2 Serialization as "application/xml"

The instance data [p.18] of the input, output or fault message is serialized as an XML document in the message body of the HTTP request, following the serialization defined in [*Canonical XML [p.36]*].

The `Content-Type` HTTP header MUST have the value `application/xml`, or a media type compatible with `application/xml`. Other HTTP headers, such as `Content-Encoding` or `Transfer-Encoding`, MAY be used.

3.8.3 Serialization as "multipart/form-data"

This format is for legacy compatibility to permit the use of XForms clients with [*IETF RFC 2388 [p.36]*] servers. This serialization format may only be used for interface operations using the Multipart Style format as defined in **3.9.2 Multipart style** [p.31] .

Each element in the sequence is serialized into a part as follow:

1. The `Content-Disposition` header MUST have the value `form-data`, and its `name` parameter is the local name of the element.

2. The `Content-Type` header MUST have the value:

   - `application/xml` (or a media type compatible with `application/xml`) if the element has a complex type;

   - `application/octet-stream` if the element is of type `xs:base64Binary`, `xs:hexBinary`, or a derived type;

   - `text/plain` if the element has a simple type; The charset MUST be set appropriately. UTF-8 or UTF-16 MUST be at least supported.

3. If the type is `xs:base64Binary`, `xs:hexBinary`, `xs:anySimpleType` or a derived type, the content of the part is the content of the element. If the type is a complex type, the element is serialized following the rules defined in the **3.8.2 Serialization as application/xml** [p.29] .

Example 3-3. Example of multipart/form-data

The following instance data of an input message:

```
<data>
  <town>
    <name>FrÃ©jus</name>
    <country>France</country>
  </town>
  <date>2004-01-16</date>
</data>
```

with the following `operation` element

```
<operation ref='t:io'
    whttp:location='temperature'
    whttp:method='POST'
    whttp:inputSerialization='multipart/form-data'/>
```

will serialize the message as follow:

```
Content-Type: multipart/form-data; boundary=AaB03x
Content-Length: xxx

--AaB03x
Content-Disposition: form-data; name="town"
Content-Type: application/xml
<town>
  <name>FrÃ©jus</name>
  <country>France</country>
</town>
--AaB03x
Content-Disposition: form-data; name="date"
Content-Type: text/plain; charset=utf-8
2004-01-16
--AaB03x--
```

3.9 Operation Styles

This section defines operation styles in use in parallel to the HTTP 1.1 binding.

3.9.1 URI Style

The URI style is selected by assigning the Interface Operation component's {style} property the value *http://www.w3.org/2004/08/wsdl/style/uri*.

The URI style may only be used for Interface Operation components whose {message exchange pattern} property has the value 'http://www.w3.org/2004/08/wsdl/in-only', 'http://www.w3.org/2004/08/wsdl/robust-in-only' or 'http://www.w3.org/2004/08/wsdl/in-out'.

Use of this value indicates that XML Schema [*XML Schema Structures [p.37]*] was used to define the schemas of the {element} properties of all Message Reference components of the Interface Operation component with {direction} property *in*. Those schemas MUST adhere to the rules below.

- The content model of input message elements are defined using a complex type that contains a sequence from XML Schema.
- The sequence MUST only contain elements. It MUST NOT contain other structures such as xs:choice.
- The sequence MUST contain only local element children. These child elements MAY contain the `nillable` attribute, and the attributes `minOccurs` and `maxOccurs` MUST have a value `0` or `1`.
- The localPart of input element's QName MUST be the same as the Interface operation component's name.
- The complex types that defines the body of an input element or its children elements MUST NOT contain any attributes.
- The input sequence MUST NOT contain multiple children element declared with the same local name.
- If the children elements of the input sequence are defined using an XML Schema type, they MUST derive from `xs:simpleType`, and MUST NOT be of the type or derive from `xs:QName`, `xs:NOTATION`, `xs:hexBinary` or `xs:base64Binary`.

3.9.2 Multipart style

The Multipart style is selected by assigning the Interface Operation component's {style} property the value *http://www.w3.org/2004/08/wsdl/style/multipart*.

The Multipart style may only be used for Interface Operation components whose {message exchange pattern} property has the value 'http://www.w3.org/2004/08/wsdl/in-only', 'http://www.w3.org/2004/08/wsdl/robust-in-only' or 'http://www.w3.org/2004/08/wsdl/in-out'.

Use of this value indicates that XML Schema [*XML Schema Structures [p.37]*] was used to define the schemas of the {element} properties of all Message Reference components of the Interface Operation component with {direction} property *in*. Those schemas MUST adhere to the rules below.

- The content model of input message elements are defined using a complex type that contains a sequence from XML Schema.

- The sequence MUST only contain elements. It MUST NOT contain other structures such as xs:choice.
- The sequence MUST contain only local element children. These child elements MAY contain the `nillable` attribute, and the attributes `minOccurs` and `maxOccurs` MUST have a value `1`.
- The localPart of input element's QName MUST be the same as the Interface operation component's name.
- The complex types that defines the body of an input element or its children elements MUST NOT contain any attributes.
- The input sequence MUST NOT contain multiple children element declared with the same local name.

3.10 Specifying the transfer coding

3.10.1 Description

Every Binding Message Reference component MAY indicate which transfer codings, as defined in section 3.6 of [*IETF RFC 2616 [p.36]*], are available for this particular message.

The HTTP binding provides a mechanism for indicating a default value at the Binding component and Binding Operation levels.

If no value is specified, no claim is being made.

3.10.2 Relationship to WSDL Component Model

The HTTP binding specification adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {http transfer coding}, a string value to the Binding Message Reference component.

3.10.3 XML Representation

```
<definitions >
  <binding name="xs:NCName" interface="xs:QName"? type="xs:anyURI"
           whttp:defaultTransferCoding="xs:string"? >
    <operation location="xs:anyURI"?
               whttp:defaultTransferCoding="xs:string" ? >
      <input messageLabel="xs:NCName"?
             whttp:transferCoding="xs:string"? />

      <output messageLabel="xs:NCName"?
              whttp:transferCoding="xs:string"? />
    </operation>
  </binding>
</definitions>
```

The XML representation for specifying the default transfer coding is an OPTIONAL *attribute information item* for the `binding` *element information item* or `binding` 's child `operation` *element information item*s with the following Infoset properties:

- A [local name] of `defaultTransferCoding`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
- A type of *wsdls:string*
- No default value

The XML representation for specifying the transfer coding is an OPTIONAL *attribute information item* with the following Infoset properties:

- A [local name] of `transferCoding`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
- A type of *wsdls:string*
- No default value

3.10.4 Mapping Between Component Properties and XML Representation

See Table 3-5 [p.33] .

Table 3-5. Mapping between Message Reference Component Extension Properties and XML Representation

Property	Mapping
{http transfer coding}	The actual value of the `whttp:transferCoding` *attribute information item* on the Binding Message Reference component, if present. If not, the actual value of the `whttp:defaultTransferCoding` on the Binding Operation component, if present. If not, the actual value of the `whttp:defaultTransferCoding` on the Binding component, if present.

3.11 Specifying the Use of HTTP Cookies

3.11.1 Description

Every Binding component MAY indicate whether HTTP cookies (as defined by [*IETF RFC 2965 [p.37]*]) are used for some or all of operations of the interface that this binding applies to.

3.11.2 Relationship to WSDL Component Model

The HTTP binding specification adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {http cookies}, a boolean value to the Binding component.

3.11.3 XML Representation

```
<definitions >
  <binding name="xs:NCName" interface="xs:QName"? type="xs:anyURI"
           whttp:cookies="xs:boolean"? >
  </binding>
</definitions>
```

The XML representation for specifying the use of HTTP cookies is an OPTIONAL *attribute information item* with the following Infoset properties:

- A [local name] of `cookies`
- A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
- A type of *wsdls:boolean*
- A default value of *false*

3.11.4 Mapping Between Component Properties and XML Representation

See Table 3-6 [p.34] .

Table 3-6. Mapping between Binding Component Extension Properties and XML Representation

Property	Mapping
{http cookies}	The actual value of the `whttp:cookies` *attribute information item*.

3.12 Specifying HTTP Access Authentication

3.12.1 Description

Every Endpoint component MAY indicate the use of an HTTP access authentication mechanism (as defined by [*IETF RFC 2616 [p.36]*]) for the endpoint described.

This binding specification allows the authentication scheme and realm to be specified.

3.12.2 Relationship to WSDL Component Model

The HTTP binding specification adds the following property to the WSDL component model (as defined in [*WSDL 2.0 Core Language [p.37]*]):

- {http authentication scheme}, a string value to the Endpoint component, corresponding to the HTTP authentication scheme used. The valid values are "basic" for the "basic" authentication scheme defined in [*IETF RFC 2617 [p.36]*], "digest" for the Digest Access Authentication scheme as defined in [*IETF RFC 2617 [p.36]*], and "none" for no access authentication.

- {http authentication realm}, a string value to the Endpoint. It corresponds to the realm authentication parameter defined in [*IETF RFC 2617 [p.36]*]. If the the value of the {http authentication scheme} property is not "none", it MUST not be empty.

3.12.3 XML Representation

```
<definitions>
  <service>
    <endpoint name="xs:NCName" binding="xs:QName" address="xs:anyURI"? >
              whttp:authenticationType="xs:string"?
              whttp:authenticationRealm="xs:string"? />
    </endpoint>
  </service>
</definitions>
```

The XML representation for specifying the use of HTTP access authentication is two OPTIONAL *attribute information item*s with the following Infoset properties:

- An OPTIONAL `authenticationType` *attribute information item* with the following Infoset properties:
  - A [local name] of `authenticationType`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
  - A type of *wsdls:string*
  - A default value of *"none"*
- An OPTIONAL `authenticationType` *attribute information item* with the following Infoset properties:
  - A [local name] of `authenticationRealm`
  - A [namespace name] of "http://www.w3.org/2004/08/wsdl/http"
  - A type of *wsdls:string*

- A default value of ""

3.12.4 Mapping Between Component Properties and XML Representation

See Table 3-7 [p.36] .

Table 3-7. Mapping between Endpoint Component Extension Properties and XML Representation

Property	Mapping
{http authentication scheme}	The actual value of the `whttp:authenticationType` *attribute information item*.
{http authentication realm}	The actual value of the `whttp:authenticationRealm` *attribute information item*.

4. References

4.1 Normative References

[Canonical XML]
Canonical XML, J. Boyer, Author. World Wide Web Consortium, 15 March 2001. This version of the Canonical XML Recommendation is http://www.w3.org/TR/2001/REC-xml-c14n-20010315. The latest version of Canonical XML is available at http://www.w3.org/TR/xml-c14n.

[IETF RFC 2119]
Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2119.txt.

[IETF RFC 2388]
Returning Values from Forms: multipart/form-data, L. Masinter, Author. Internet Engineering Task Force, August 1998. Available at http://www.ietf.org/rfc/rfc2388.txt.

[IETF RFC 2396]
Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, Authors. Internet Engineering Task Force, August 1998. Available at http://www.ietf.org/rfc/rfc2396.txt.

[IETF RFC 2616]
Hypertext Transfer Protocol -- HTTP/1.1, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Authors. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2616.txt.

[IETF RFC 2617]
HTTP Authentication: Basic and Digest Access Authentication, J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, June 1999. Available at http://www.ietf.org/rfc/rfc2616.txt.

[IETF RFC 2818]
HTTP Over TLS, E. Rescorla, Author. Internet Engineering Task Force, May 2000. Available at http://www.ietf.org/rfc/rfc2818.txt.

[IETF RFC 2965]
HTTP State Management Mechanism, D. Kristol, L. Montulli Authors. Internet Engineering Task Force, October 2000. Available at http://www.ietf.org/rfc/rfc2965.txt.

[IETF RFC 3023]
XML Media Types, M. Murata, S. St. Laurent, D. Kohn, Authors. Internet Engineering Task Force, January 2001. Available at http://www.ietf.org/rfc/rfc3023.txt.

[XForms 1.0]
XForms 1.0, M. Dubinko, et al., Editors. World Wide Web Consortium, 14 October 2003. This version of the XForms 1.0 Recommendation is http://www.w3.org/TR/2003/REC-xforms-20031014/. The latest version of XForms 1.0 is available at http://www.w3.org/TR/xforms/.

[SOAP 1.2 Part 1: Messaging Framework]
SOAP Version 1.2 Part 1: Messaging Framework, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielsen, Editors. World Wide Web Consortium, 24 June 2003. This version of the "SOAP Version 1.2 Part 1: Messaging Framework" Recommendation is http://www.w3.org/TR/2003/REC-soap12-part1-20030624/. The latest version of "SOAP Version 1.2 Part 1: Messaging Framework" is available at http://www.w3.org/TR/soap12-part1/.

[SOAP 1.2 Part 2: Adjuncts]
SOAP Version 1.2 Part 2: Adjuncts, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, and H. Frystyk Nielsen, Editors. World Wide Web Consortium, 7 May 2003. This version of the "SOAP Version 1.2 Part 2: Adjuncts" Recommendation is http://www.w3.org/TR/2003/REC-soap12-part2-20030624/. The latest version of "SOAP Version 1.2 Part 2: Adjuncts" is available at http://www.w3.org/TR/soap12-part2/.

[XML 1.0]
Extensible Markup Language (XML) 1.0 (Second Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Editors. World Wide Web Consortium, 10 February 1998, revised 6 October 2000. This version of the XML 1.0 Recommendation is http://www.w3.org/TR/2000/REC-xml-20001006. The latest version of XML 1.0 is available at http://www.w3.org/TR/REC-xml.

[XML Information Set]
XML Information Set, J. Cowan and R. Tobin, Editors. World Wide Web Consortium, 24 October 2001. This version of the XML Information Set Recommendation is http://www.w3.org/TR/2001/REC-xml-infoset-20011024. The latest version of XML Information Set is available at http://www.w3.org/TR/xml-infoset.

[XML Schema Structures]
XML Schema Part 1: Structures, H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 1 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-1-20010502. The latest version of XML Schema Part 1 is available at http://www.w3.org/TR/xmlschema-1.

[XML Schema Datatypes]
XML Schema Part 2: Datatypes, P. Byron and A. Malhotra, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 2 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-2-20010502. The latest version of XML Schema Part 2 is available at http://www.w3.org/TR/xmlschema-2.

[WSDL 2.0 Core Language]
Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, R. Chinnici, M. Gudgin, J-J. Moreau, S. Weerawarana, Editors. World Wide Web Consortium, 3 August 2004. This version of the "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language"

Specification is available is available at http://www.w3.org/TR/2004/WD-wsdl20-20040803. The latest version of "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language" is available at http://www.w3.org/TR/wsdl20.

[WSDL 2.0 Predefined Extensions]
Web Services Description Language (WSDL) Version 2.0 Part 2: Predefined Extensions, M. Gudgin, A. Lewis, and J. Schlimmer, Editors. World Wide Web Consortium, 3 August 2004. This version of the "Web Services Description Language (WSDL) Version 2.0 Part 2: Predefined Extensions" Specification is available at http://www.w3.org/TR/2004/WD-wsdl20-extensions-20040803. The latest version of "Web Services Description Language (WSDL) Version 2.0 Part 2: Predefined Extensions" is available at http://www.w3.org/TR/wsdl20-extensions.

4.2 Informative References

[SOAP Message Transmission Optimization Mechanism]
SOAP Message Transmission Optimization Mechanism, N. Mendelsohn, M. Nottingham, and H. Ruellan, Editors. World Wide Web Consortium, Working Draft 8 June 2004.

[MTXML]
Assigning Media Types to Binary Data in XML, A. Karmarkar, Ã. YalÃ§Ä±nalp, W3C Working Draft, 8 June 2004. The latest version of the "Assigning Media Types to Binary Data in XML" document is available from http://www.w3.org/TR/xml-media-types/.

[WSDL 2.0 Primer]
Web Services Description (WSDL) Version 2.0: Primer, K. Sankar, K. Liu, D. Booth, Editors. World Wide Web Consortium. The editors' version of the "Web Services Description Version 2.0: Primer" document is available from http://dev.w3.org/cvsweb/~checkout~/2002/ws/desc/wsdl20/wsdl20-primer.html.

[XML 1.1]
Extensible Markup Language (XML) 1.1 , T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, Francois Yergau, and John Cowan, Editors. World Wide Web Consortium, 04 February 2004, edited in place 15 April 2004. This version of the XML 1.1 Recommendation is http://www.w3.org/TR/2004/REC-xml-20040204. The latest version of XML 1.1 is available at http://www.w3.org/TR/xml11.

A. Acknowledgements (Non-Normative)

This document is the work of the W3C Web Service Description Working Group.

Members of the Working Group are (at the time of writing, and by alphabetical order): David Booth (W3C), Allen Brookes (Rogue Wave Softwave), Helen Chen (Agfa-Gevaert N. V.), Roberto Chinnici (Sun Microsystems), Ugo Corda (SeeBeyond), Glen Daniels (Sonic Software), Paul Downey (British Telecommunications), Youenn Fablet (Canon), Martin Gudgin (Microsoft Corporation), Hugo Haas (W3C), Hao He (The Thomson Corporation), Tom Jordahl (Macromedia), Jacek Kopecky (Digital Enterprise Research Institute (DERI)), Amelia Lewis (TIBCO Software, Inc.), Kevin Canyang Liu (SAP), Jonathan Marsh (Microsoft Corporation), Peter Madziak (Agfa-Gevaert N. V.), Josephine Micallef (SAIC - Telcordia Technologies), Jeff Mischkinsky (Oracle Corporation), Dale Moberg (Cyclone Commerce), Jean-Jacques Moreau (Canon), Mark Nottingham (BEA Systems, Inc.), David Orchard (BEA Systems, Inc.), Bijan Parsia (University of Maryland), Arthur Ryman (IBM), Adi Sakala (IONA Technologies),

Jeffrey Schlimmer (Microsoft Corporation), Igor Sedukhin (Computer Associates), Jerry Thrasher (Lexmark), William Vambenepe (Hewlett-Packard Company), Asir Vedamuthu (webMethods, Inc.), Sanjiva Weerawarana (IBM), Ãmit YalÃ§Ä±nalp (Oracle Corporation), Prasad Yendluri (webMethods, Inc.).

Previous members were: Lily Liu (webMethods, Inc.), Don Wright (Lexmark), Joyce Yang (Oracle Corporation), Daniel Schutzer (Citigroup), Dave Solo (Citigroup), Stefano Pogliani (Sun Microsystems), William Stumbo (Xerox), Stephen White (SeeBeyond), Barbara Zengler (DaimlerChrysler Research and Technology), Tim Finin (University of Maryland), Laurent De Teneuille (L'Echangeur), Johan Pauhlsson (L'Echangeur), Mark Jones (AT&T), Steve Lind (AT&T), Sandra Swearingen (U.S. Department of Defense, U.S. Air Force), Philippe Le HÃ©garet (W3C), Jim Hendler (University of Maryland), Dietmar Gaertner (Software AG), Michael Champion (Software AG), Don Mullen (TIBCO Software, Inc.), Steve Graham (Global Grid Forum), Steve Tuecke (Global Grid Forum), Michael Mahan (Nokia), Bryan Thompson (Hicks & Associates), Ingo Melzer (DaimlerChrysler Research and Technology), Sandeep Kumar (Cisco Systems), Alan Davies (SeeBeyond), Jacek Kopecky (Systinet), Mike Ballantyne (Electronic Data Systems), Mike Davoren (W. W. Grainger), Dan Kulp (IONA Technologies), Mike McHugh (W. W. Grainger), Michael Mealling (Verisign), Waqar Sadiq (Electronic Data Systems), Yaron Goland (BEA Systems, Inc.).

The people who have contributed to discussions on www-ws-desc@w3.org are also gratefully acknowledged.

B. Part 3 Change Log (Non-Normative)

Date	Author	Description
20040730	HH	Removed `property` on `wsoap:module` in pseudo-schema.
20040730	HH	Removed AD Feature HTTP serialization.
20040729	HH	Added AD Feature support in HTTP binding.
20040727	HH	Clarified interaction between SOAP binding and HTTP binding properties
20040727	HH	Renamed http prefix whttp
20040727	SW	Implemented Umit's proposal to mark MTOM as one optimization mechanism.
20040726	HH	Restricted URI style with regards to QNames and added trailing / in URL-encoded syntax
20040723	HH	Addressed issue 246: limited MEP to In-Out, In-Only and Robust In-Only
20040723	HH	Addressed issue 226.
20040723	HH	Addressed 249: major reorganization of the HTTP binding to be presented in a functional way like the SOAP binding rather than in a syntactical way.

20040722	SW	Moved SOAP binding syntax summary to the top per request. Also fixed the value of the binding/@type property in the pseudo-schema to show that its a SOAP binding.
20040722	HH	Added HTTP error code attribute on fault binding. Added relationship between instance data and properties in the component model. Addresses issue 166.
20040722	HH	Renamed SOAP protocol into underlying protocol.
20040721	HH	Set the {type} property of binding for HTTP binding.
20040721	HH	Fixes for issue 177.
20040720	HH	Cross-referenced Part 1 properties.
20040720	HH	Specified default serialization format for HTTP binding, as well as made clear how the defined serialization formats apply constraints on interface operation styles
20040705	JJM	Added note to indicate only one element per SOAP body.
20040702	SW	Corrected how the SOAP binding is indicated .. I had forgotten about binding/@type!
20040625	SW	Made pseudo-syntax consistent with part1
20040624	SW	Update the rest of the SOAP binding stuff and consistified everything.
20040624	SW	Cleaned up how SOAP modules were described. Added default SOAP MEP stuff.
20040623	SW	Added default binding rules about HTTP URI generation.
20040623	SW	Added default binding rules about SOAP MEP selection and HTTP Method selection.
20040623	SW	Fixed up soapaction default rules
20040623	SW	Allowed use of MTOM for payload serialization
20040623	SW	Fixed up the wsoap:protocol section
20040618	SW	Re-introduced AII and EII entity refs.
20040618	SW	Made soap:module compose with nearest-wins rule.
20040606	DO	Cleanup on http binding section - had missed some properties. completed removal of @separator
20040604	DO	Major rewrite of http binding. Moved to component model, added http properties, added input/output serialization, removed @separator, added self as editor
20040526	SW	Removed wsoap:address
20040526	SW	Editorial/small corrections per F2F decisions

20040526	SW	Made soap binding be mostly attribute based per F2F decision
20040519	SW	removed spurious fault element inside binding/operation/{in,out}put from syntax summary
20040519	SW	Put in wsoap:module at operation level in the syntax summary (was missing)
20040519	SW	Removed old SOAP binding text
20040519	SW	Removed wsoap:header
20040519	JJM	Added SOAP Address section
20040519	JJM	Added SOAP Operation section
20040519	JJM	Replace reference to "XML" by "XML1.0"
20040519	JJM	Added SOAP Fault section
20040519	JJM	Added SOAP Header section
20040519	JJM	Added SOAP Module section
20040516	SW	Finished writing up soap:binding
20040516	SW	Added myself as an editor.
20040514	SW	Added default binding rules.
20040514	SW	Commented out old totally out of date SOAP binding.
20040514	JJM	Rework the binding and module sections. Reindent to match the structure of the HTTP binding.
20040511	JJM	Updated SOAP binding pseudo-schema, according to telcon 20040506.
20040511	JJM	Updated SOAP binding introduction.
20040401	JJM	Fixed one remaining occurrence of "verb" (instead of "method").
20040326	JJM	Sanitized ednotes. Added new ednotes indicating the SOAP binding needs work and the HTTP binding is (mostly) OK.
20040326	JJM	Added Philippe’s note on URIPath, as per telcon 20040325.
20040305	JJM	Removed the archaic MIME binding, now superseeded by the HTTP binding anyway.
20040305	JJM	Included Philippe’s changes to the HTTP binding.
20031103	JJM	Fix new non-normative SOAP binding pseudo-schema.
20031102	SW	Updated SOAP binding.
20031102	SW	Change 1.2 to 2.0 per WG decision to rename.

20030606	JJM	Replaced <kw/> by <b/>. Indicated that pseudo-schemas are not normative
20030604	JJM	Reformated pseudo-syntax elements to match Part 1 layout
20030529	JCS	Incorporated text to resolve Issue 6e
20030523	JJM	Commented out MIME binding example; this is primer stuff.
20030523	JJM	Added pseudo-syntax to all sections.
20030523	JJM	Started converting the fault and headerfault sections to component model.
20030523	JJM	Complete the Multipart and x-www-form-urlencoded sections.
20030523	JJM	Fixed typos in HTTP binding (in particular added NOT in some section headers).
20030522	JCS	Added rules for serializing HTTP response
20030522	JCS	Added cardinality to pseudo schema for HTTP binding
20030522	JCS	Changes @transport to @protocol for SOAP binding
20030522	JJM	Incorporated remaining text from Philippe into the HTTP binding.
20030522	JJM	Polished the HTTP binding, split into subsections, added double curly brace escape mechanism, removed pseudo-schema.
20030521	JCS	Added rules for @verbDefault/@verb and @location.
20030514	JJM	Start converting the HTTP binding to the component model. The next thing to do will be to remove http:urlReplacement, etc. and incorporate instead Philippe's text.
20030313	MJG	Changed to Part 3 (from Part 2)
20030117	JCS	Incorporated resolution for Issue 5 (@encodingStyle). Referenced (rather than in-lined XML Schema).
20030117	JJM	Various editorial fixes.
20030116	JCS	Updated pseudo and XML Schema.
20030116	JJM	Added propertyConstraint section.
20030116	JJM	Added soap:module section.
20030115	JCS	Incorporated resolutions for Issue 25 (drop @use and @encoding), Issue 51 (headers reference element/type), and attribute roll up into text and schema. Began reworking SOAP HTTP binding to use Infoset model. Removed informative appendices 'Notes on URIs' and example WSDL documents; expect them to appear in the primer. Updated SOAP 1.2 references to CR.
20030114	JJM	Removed ednote saying Part 2 is out of synch with Part 1.
20030111	JJM	Incorporated resolution for issue 17 (role AII).

20030109	JJM	Incorporated resolution for issue 4 (Namespaces).
20020702	JJM	Added summary to prefix table.
20020628	JJM	Added out-of-synch-with-Part2 and not-soap12-yet ednote.
20020621	JJM	Commented out the link to the previous version. There is no previous version for 1.2 right now.
20020621	JJM	Rewrote the Notation Conventions section.
20020621	JJM	Added reference to part 0 in introduction. Renumbered references.
20020621	JJM	Simplified abstract and introduction.
20020621	JJM	Obtain the list of WG members from a separate file.
20020621	JJM	Updated stylesheet and DTDs to latest XMLP stylesheet and DTDs.
20020621	JJM	Deleted placeholder for appendix C "Location of Extensibility Elements", since this is part 1 stuff and extensibility has been reworked anyway.
20020621	JJM	Corrected link to issues lists
20020621	JJM	Updated title from "WSDL" to "Web Services Description Language". Now refer to part 1 as "Web Services... Part 1: Framework
20020621	JJM	Added Jeffrey as an editor :-). Removed Gudge (now on Part 2) :-(
20020411	JJM	Fixed typos noticed by Kevin Liu
20020301	JJM	Converted the "Schemas" sections
20020301	JJM	Converted the "Wire WSDL examples" sections
20020301	JJM	Converted the "Notes on URIs" sections
20020301	JJM	Converted the "Notational Conventions" sections
20020301	JJM	Converted the "References" sections
20020301	JJM	Converted the "MIME Binding" section to XML
20020221	JJM	Converted the "HTTP Binding" section to XML
20020221	JJM	Added placeholders for the "Wire examples" and "Schema" sections
20020221	JJM	Converted the "SOAP Binding" section to XML
20020221	JJM	Added the Change Log
20020221	JJM	Added the Status section
20020221	JJM	Simplified the introduction; referred to Part1 for a longer introduction

20020221	JJM	Renamed to "Part 2: Bindings"
20020221	JJM	Created from http://www.w3.org/TR/2001/NOTE-wsdl-20010315